

# Attunity Connect Guide

Version 4.1



## ***Attunity Connect Guide***

*© 2003 by Attunity Ltd.*

Due to a policy of continuous development, Attunity Ltd. reserves the right to alter, without prior notice, the specifications and descriptions outlined in this document. No part of this document shall be deemed to be part of any contract or warranty whatsoever.

Attunity Ltd. retains the sole proprietary rights to all information contained in this document. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopy, recording, or otherwise, without prior written permission of Attunity Ltd. or its duly appointed authorized representatives.

Product names mentioned in this document are for identification purposes only and may be trademarks or registered trademarks of their respective companies.

### **3rd party software credits**

Attunity products (Attunity Connect and Attunity Connect Studio) include software developed by Eclipse.org, Exolab.org, Sun Microsystems, Inc., the JDOM project (<http://www.jdom.org/>) and the Apache Software Foundation (<http://www.apache.org/>).

Attunity hereby disclaims on behalf of all Eclipse.org contributors whose components are included in Attunity Connect, all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability. Attunity excludes on behalf of all Eclipse.org contributors whose components are included in Attunity Connect all liability for damages, including direct, indirect, special, incidental and consequential damages.

Attunity hereby agrees to defend and indemnify Sun and its licensors from and against any damages, costs, liabilities, settlement amounts and/or expenses (including attorneys' fees) incurred in connection with any claim, lawsuit or action by any third party that arises or results from the use or distribution of any and all Programs and/or Software, as related to Sun's software components embedded in this software (Attunity Connect).

Castor is Copyright 2000-2002 (C) Intalio Inc. All Rights Reserved.

JDOM is Copyright (C) 2000-2002 Brett McLaughlin & Jason Hunter. All rights reserved.

The Apache Software License, Version 1.1 is Copyright (c) 1999-2000 The Apache Software Foundation. All rights reserved.

# Table of Contents

<b>About this Manual</b>	<b>7</b>
<b>Chapter 1: What is Attunity Connect?</b>	<b>9</b>
The Attunity Connect Application Engine .....	10
The Attunity Connect Data Engine .....	12
Base Services.....	17
Supported Platforms .....	21
<b>Chapter 2: A Quick Walkthrough – Accessing Data</b>	<b>23</b>
Setting Up Access to a Client Machine to Access Data .....	23
Setting Up Access to a Machine .....	23
Setting Up Access to Data .....	26
Testing the Connection.....	28
Setting Up Access to Data on a Server Machine .....	30
Testing the Data Source Shortcut.....	31
Connecting to Multiple Data Sources in a Single SQL Statement .....	32
<b>Chapter 3: Setting Up Access to Data and Applications</b>	<b>33</b>
Configuring Access to Machines with Data or Applications.....	35
Using an Offline Design Machine to Create Attunity Connect Definitions.....	37
Creating a New Binding Configuration.....	37
Setting the Binding Environment.....	39
Setting Up Access to Other Machines.....	40
Adding a Data Source .....	41
Adding a Data Source Shortcut.....	42
Adding an Adapter .....	45
Adding an Event.....	46
<b>Chapter 4: Configuring Client/Server Communication</b>	<b>47</b>
Starting the Daemon.....	48
Runtime Manager Perspective .....	50
Configuring a Daemon .....	55
Editing the Daemon.....	56

<b>Chapter 5: Metadata for a Data Source</b>	<b>67</b>
Importing Metadata to Attunity Connect .....	68
Managing Attunity Connect Metadata .....	70
Updating Statistics .....	73
Creating Procedure Metadata .....	75
<b>Chapter 6: Metadata for an Application Adapter</b>	<b>79</b>
Importing Adapter Metadata .....	79
Managing Metadata with Attunity Studio .....	80
<b>Chapter 7: Metadata for an Event</b>	<b>85</b>
Creating Metadata .....	85
Managing Metadata with Attunity Studio .....	87
<b>Chapter 8: Setting Up Security</b>	<b>93</b>
Configuring Design Time Security .....	93
Initially Accessing the Machine .....	94
User Authorization Within Attunity Studio.....	95
Workspace Authorization Within Attunity Studio.....	96
User Profile Security.....	96
Configuring Runtime Security .....	96
Daemon and Workspace Administration Rights .....	97
Client Access .....	99
Passing Through a Firewall .....	102
Encrypting Network Communications .....	103
<b>Chapter 9: The Connect String From an Application to Attunity Connect</b>	<b>107</b>
The JDBC Connect String .....	107
The ODBC Connect String .....	108
The ADO Connect String .....	111
The XML Connection Document .....	111
The JCA Connection .....	112
The COM Connection.....	114

<b>Chapter 10: Using Attunity Connect SQL Enhancements</b>	<b>115</b>
Batching SQL Statements.....	115
Joining Data from Multiple Data Sources in a Single Query.....	116
Hierarchical Queries.....	116
Generating Hierarchical Results Using SQL.....	117
Accessing Hierarchical Data Using SQL.....	118
Flattening Hierarchical Data Using SQL.....	118
Using Virtual Tables to Represent Hierarchical Data.....	119
Copying Data From One Table to Another.....	120
Passthru SQL.....	121
<b>Chapter 11: Using the Demo ADO Application</b>	<b>123</b>
Connecting to Attunity Connect via ChapView .....	124
Specifying and Executing Queries .....	124
Working with Parameterized Queries .....	125
Modifying Data in a Recordset .....	126
Working with Chapters .....	127
Modifying Chapters .....	127
Specifying Recordset Properties.....	128
Working with Schemas.....	129

## Appendixes

<b>Appendix A: Importing Metadata for a Data Source</b>	<b>131</b>
<b>Appendix B: Importing Metadata for an Adapter</b>	<b>143</b>
<b>Appendix C: Generating Metadata for a Database Adapter</b>	<b>151</b>
Modifying the Interactions .....	157



# About this Manual

Attunity Connect is an information infrastructure solution that provides built-in connectivity to data sources and applications across a distributed environment that can encompass different platforms, networks, and even the Internet.

In addition to a range of ready-to-use drivers, adapters and integration software, Attunity Connect includes development and management tools that make it easy to add applications and data sources to the integrated enterprise.

Attunity Connect provides a flexible, dependable infrastructure for web-enabling mainframe and other legacy solutions. The architecture takes advantage of distributed processing and an array of industry standards (such as XML, JCA, COM, JDBC, ODBC and OLE DB).

This manual describes the tasks required to configure and manage Attunity Connect.

In addition, a tutorial guides you through setting up and using Attunity Connect to access data on a client and server machine, in order to understand principles of using the product.

Full reference material is available in *Attunity Connect Reference*.

## Intended Audience

This manual is intended for users of Attunity Connect who are:

- System Administrators and those responsible for access to the databases and applications at the site.
- Developers writing applications that will use Attunity Connect to access data and other applications.

## Related Information

This manual assumes you are familiar with whichever of the following you use to access data:

- Sun's JDBC data access specification
- The Microsoft ODBC data access specification
- The Microsoft ActiveX Data Objects (ADO) specification
- Standard ANSI '92 SQL
- Sun's J2EE Connector Architecture (JCA)
- XML

**Typographic Conventions**

This document uses the following typographic conventions:

<i>Italics</i>	Italics denotes either a placeholder/variable for an actual value or a new term.
► <b>To do</b>	A bolded sentence starting with the word <b>To</b> is followed by a procedure.
❖	This symbol prefaces a note to the main text.
<code>Sample code</code>	This font denotes sample code and commands that you type on the keyboard.
<b>Xyz Platforms</b>	Grey shading highlights platform-specific information within a topic.

**Conventions for Commands**

The following conventions are used to describe commands entered to the keyboard or console:

- Keywords are shown in standard type.
- Variables are shown as italics.
- Square brackets ([]) enclose optional items.
- Vertical lines (|) separate between choices.
- Ellipses (...) indicate that the preceding item can be repeated.

**Related Documentation**

The Attunity Connect documentation set includes the following:

<i>Attunity Connect Release Notes</i>	New features for the current Attunity Connect version.
<i>Getting Started with Attunity Connect</i>	A walk-through describing how to connect to data using Attunity Connect.
<i>Attunity Connect Installation Guides</i>	Guides describing how to install Attunity Connect on HP (Compaq) NonStop, OpenVMS, OS/390 and Z/OS, OS/400, UNIX and Windows platforms.
<i>Attunity Connect Reference</i>	A complete reference to Attunity Connect.
<i>Attunity Connect Guide</i>	This manual.
<i>Attunity Connect and Applications</i>	Tutorials demonstrating how to use Attunity Connect with various applications, such as IBM WebSphere.
<i>Using the Attunity Connect Syntax File (NAV.SYN)</i>	Documentation detailing how to use the Attunity Connect syntax file to control the way SQL is processed by Attunity Connect.
<i>Attunity Connect Developer SDK</i>	Documentation detailing the Attunity Connect SDK, which enables you to build application adapters and data adapters. In addition, you can define custom data types and start-up functions.



# Chapter 1    What is Attunity Connect?

Attunity Connect is an information infrastructure solution that provides built-in connectivity to data sources and applications across a distributed environment that can encompass different platforms, networks, and even the Internet.

In addition to a range of ready-to-implement adapters and integration software, Attunity Connect includes development and management tools that make it easy to add applications and data sources to the integrated enterprise.

Attunity Connect provides a flexible, dependable infrastructure for web-enabling mainframe and other legacy solutions. The architecture takes advantage of distributed processing and an array of industry standards (such as XML, JCA, JDBC, ODBC and OLE DB).

At the heart of the Attunity Connect integration solution are the Attunity Connect Engines. These shared, multi-platform engines manage access and updates across the IT infrastructure. They can also integrate information from disparate systems with a single request.

The Attunity Connect engines can receive requests from a wide variety of supported APIs. The relevant engine processes the requests – interacting with applications and data sources via Attunity Connect drivers and adapters – and returns the results, again via the interfaces. Attunity Connect buffers the user from the applications and data sources requested, so that it appears as one consistent API and data model, regardless of the source. The movement of information from any of the supported applications and data sources to the consumer is completely transparent to the consumer application.

The engines provide comprehensive transaction support to the extent allowed by the sources' support for two-phase commit functions.

Attunity Connect implementations include the following engines:

**The Attunity Connect Application Engine** – Accommodates complex data structures and interactions, giving web and other applications direct, efficient access to applications via Attunity Connect application adapters.

**The Attunity Connect Data Engine** – Opens enterprise data sources as relational, structured files via Attunity Connect data. In addition to the

common relational data sources, Attunity Connect provides access to hierarchical data sources, indexed sequential files, simple files, XML-based data, personal data sources, spreadsheets, and other sources of data.

Attunity Connect exposes all its services via industry-standard data and application interfaces. This approach frees developers to choose the best API for the purpose, based on the requirements of the involved applications. It also enables developers to switch to other tools, technologies, and vendors without having to redesign applications.

Attunity Connect can be installed on one machine (client or server) or distributed throughout the network in a multi-tiered architecture.

## The Attunity Connect Application Engine

The Attunity Connect application engine provides an enterprise application integration (EAI) model that enables any kind of application to interact with applications and data sources via their own native interfaces. On the client end, applications use industry-standard interfaces to communicate with the Attunity Connect application engine. In turn, the application engine communicates with specific adapters that access enterprise and application data on the server end. An XML-based schema supports precise application mapping. As a result, the application engine opens legacy applications of all types to integration with each other and with cutting-edge technologies such as Java and XML.

Because the application engine uses an EAI model, interactions with the source application or data are precise and predictable. Requesting applications can specify exactly how interactions occur. Moreover, the application engine maps data structures faithfully, facilitating access to familiar applications within new environments. The EAI model is particularly suitable for deployment over the Internet using protocols such as TCP/IP and HTTP because it allows for both stateful and stateless interactions and can batch requests to minimize network traffic.

A distinctive feature of the application engine is its ability to translate application structures, such as those typical of legacy COBOL applications, to and from XML. Web and other solutions can use the application engine to interact with both applications and data sources through a growing number of XML-based tools, as well as other application-oriented frameworks such as Sun's J2EE JCA (J2EE Connectors Architecture) and Microsoft .NET's SOAP-based interfaces.

**Frontend APIs**

Attunity Connect supports the following APIs to access applications:

**XML Application Interface** – The Attunity Connect XML application interface enables any application with an XML API to access applications and data via Attunity Connect. The XML application interface supports an XML-based protocol modeled after the JCA architecture. This protocol is both readable by people and programmatically easy to use. This protocol is exceptionally well tailored for web-based, internet-wide use, particularly in conjunction with XML transformation engines. The XML application interface is directly available (callable) on all the platforms where Attunity Connect runs. On other platforms, it is accessible via network protocols such as TCP/IP (sockets) and HTTP.

**JCA Application Interface** – The Attunity Connect JCA (J2EE Connectors Architecture) interface supports application servers based on J2EE (Java 2 Enterprise Edition) standards. It provides a robust, efficient, and reliable way to integrate Java applications with native applications.

**COM Interface** – Attunity Connect provides a COM component that enables access to Attunity Connect application adapters and services using the Attunity Connect XML protocol. The COM component provides seamless integration with various Microsoft Windows products and technologies such as IIS/ASP, Internet Explorer, Visual Basic and .Net among other Microsoft and non-Microsoft products.

**Application Adapters**

With the Attunity Connect generic application adapter, developers can incorporate applications into Attunity Connect solutions. They can wrap and leverage existing application-specific business logic, protect data integrity by writing to a data source through its original application, and trigger operational tasks, all with a single adapter that runs consistently across diverse platforms.

**Supported Application Adapters**

Attunity Connect provides the following application adapters:

- BEA WebLogic (on Windows and UNIX platforms<sup>1</sup>)
- BizTalk Server (on Windows platforms)<sup>1</sup>
- CICS (on OS/390 and z/OS platforms)
- COM applications (on Windows platforms)
- Database and Query Adapters – access to any data server supported by Attunity Connect via XML JCA or COM.

---

<sup>1</sup>: A separate product, fully integrated with Attunity Connect.

- IMS/TM (on OS/390 and z/OS platforms)
- Legacy applications
- Pathways (on HP (Compaq) NonStop Himalaya platforms)
- Tuxedo (on Windows and UNIX platforms)

## The Attunity Connect Data Engine

The Attunity Connect data engine accesses, updates, and joins enterprise information from applications and data sources as if they were all relational databases. At the same time, it takes advantage of its query optimizer to determine the fastest way to carry out these tasks, minimizing the load on IT resources, networks, and systems.

Because the data engine uses a relational model, it normalizes the data, converting hierarchical structures into tables without redundant data. By combining the relational model with the SQL language, the data engine allows applications to issue the same complex query to multiple data sources without tuning it to each target source.

The relational approach also simplifies access via commercial tools and applications that interoperate with relational sources. Clients can use industry-standard JDBC, ODBC, and ADO/OLE DB interfaces or XML to submit SQL requests to the data engine.

When the data engine receives and parses an SQL request, it first determines which data sources are involved, where the data resides, and how the source handles data. The data engine determines how to carry out the process based on metadata that it retrieves from a local cache, from the Attunity Connect repository, or dynamically from the backend data sources. Then, the data engine generates a query execution plan in the form of a tree.

Whenever possible the data engine passes the entire request to the underlying data source. In this case, the engine translates between standard ANSI SQL '92 and the underlying databases' SQL dialect. The data engine can also accept pass-through queries to nonstandard SQL functions supported by the target source.

If a data source offers limited SQL capabilities, the data engine implements missing functions as needed. If the data source offers no SQL capabilities at all, the data engine breaks the request into simple retrieval operations that an indexed or sequential table can read.

The data engine can also work with data via generic drivers (such as ODBC and OLE DB). In this case, it uses an external syntax definition to determine which queries or parts of queries the database can handle and which it will execute itself.

**Attunity Connect  
Query Optimizer**

The data engine includes the query optimizer, that minimizes execution time and resource consumption. The query optimizer enhances the data engine's initial query execution plan based on the query structure, network structures, the target data sources' capabilities and locations, and the statistical information available for each table.

To maximize the efficiency of query execution, the query optimizer uses various caching and access techniques, including read-ahead, parallelism, and lookup-, hash-, and semi-joins. It flattens views, breaks out and propagates simple predicates down the tree, reorders joins, directs join strategies, selects indexes, and performs other related tasks.

If the target data sources are distributed across multiple machines, the data engine and query optimizer together generate a distributed execution plan that minimizes network traffic and round-trips.

**Performance Tuning Tools**

Database administrators can review and control the optimization strategies that the Attunity Connect optimizer uses. Using a query analyzer, IT staff can monitor accumulated statistics and heuristic information to evaluate the success of the optimization strategies. These tools enable users to evaluate and understand the way specific queries work by specifying hints, flags, optimization goals (first-row or all-rows optimization), and other query properties, such as requests for scrollable or updateable cursors.

**Frontend APIs**

Attunity Connect supports the following APIs to access data:

**JDBC Interface** – Attunity Connect provides a pure-Java Type-3 driver that supports J2EE JDBC (such as data sources, distributed transactions, hierarchical record sets, and connection pooling). The Attunity Connect JDBC interface is available on all platforms that support Java.

**ODBC Interface** – The Attunity Connect ODBC interface enables organizations to use the API of choice for most popular client-server business intelligence tools. The ODBC interface implements the ODBC 2.5 and ISO CLI standards, so that COBOL and other 3GL programs on any platform can call it. The Attunity Connect ODBC interface is available on all platforms running Attunity Connect.

**OLE DB/ADO Interface** – Attunity Connect provides an OLE DB/ADO interface that supports advanced features, including chapters, scrollability, and multi-threading. The OLE DB/ADO interface is compatible with all Microsoft tools and applications. This provider also functions as a database gateway for Microsoft SQL Server, allowing

SQL Server users to access all data sources available via Attunity Connect. The Attunity Connect OLE DB/ADO interface is available on the Microsoft Windows platforms.

#### **Data Source Drivers**

Native Attunity Connect data drivers utilize the native API of each data source. Attunity tailors these drivers to the individual data model and performance characteristics of the particular data source. These source-specific adapters share common logic.

These drivers deal with metadata, describing the information offered by the data sources, and with mapping the underlying data model and functionality into relational or ISAM (Index Sequential Access Methods) models. Drivers for nonrelational data sources support sequential and indexed access, array structures and hierarchical structures, and other functions without normalization or other time-consuming operations.

Drivers fall into one of the following classes:

- RDBMS drivers for data sources that:
  - Support some dialect of SQL.
  - Are capable of performing joins (and therefore work with Command objects).
- File system drivers for data sources that:
  - Do not perform joins.
  - Work with one table at a time (and therefore open Rowset objects directly on base tables).

The functionality within each class can be factored further. For RDBMS drivers, Attunity Connect needs to be aware of the syntactic and semantic nuances of the SQL dialect for each specific data source, of the library functions supported by the data source, and other details. Similarly, a file system may or may not support functionality such as indexes, BLOBs and embedded rowsets, or be capable of efficiently executing single-table predicates (filters).

For relational data sources not supported by Attunity Connect through a custom driver, Attunity Connect provides generic ODBC and OLE DB gateways. A syntax file (NAV.SYN) encapsulates backend peculiarities and facilitates the connection of the new data source to Attunity Connect via one of these generic drivers.

## Supported Data Sources

Attunity Connect provides access to the following data sources:

Relational Data Sources	Non-Relational Data Sources	Other Data Sources
DB2	Adabas	Flat Files
Informix	Btrieve	Non-SQL OLE DB data
Ingres	CISAM	Text Files
Ingres II	DISAM	
ODBC data	Enscribe	
Oracle	IMS/DB	
Oracle Rdb	Oracle Codasyl DBMS	
Red Brick	RMS	
Relational OLE DB data	VSAM	
SQL/MP		
SQL Server		
Sybase		

In addition, procedure drivers are provided to enable access to program functionality. These procedures include a generic procedure driver and a specific driver for Natural programs on an OS/390 or z/OS platform.

### Data Dictionary Editor and Metadata Utilities

The Attunity Studio Metadata perspective enables users to view, create, modify, and manage metadata for data sources, such as flat files, that require Attunity Connect metadata (ADD).

Using Attunity Studio you can also obtain statistical information for a table, which is then used to improve performance when executing a query.

In addition, a number of import utilities are provided, which enable creating metadata in Attunity Connect automatically.

**Distributed Transactions Support**

To ensure the integrity of simultaneous updates to multiple data sources, Attunity Connect supports distributed transactions in the following ways:

- As a distributed transaction manager, for safe, reliable multi-server transactions.
- As an OLE Transaction resource manager, allowing all Attunity Connect-enabled data sources to participate in distributed Microsoft MTS transactions using the OLE Transactions protocol.
- As an XA resource manager.
- Using the JDBC 2.x Interface, so that all Attunity Connect-enabled data sources can participate in distributed J2EE transactions under a J2EE application server.

A transactions log file backs up Attunity Connect two-phase-commit (2PC) operations in the case of a failure to recover transactions.

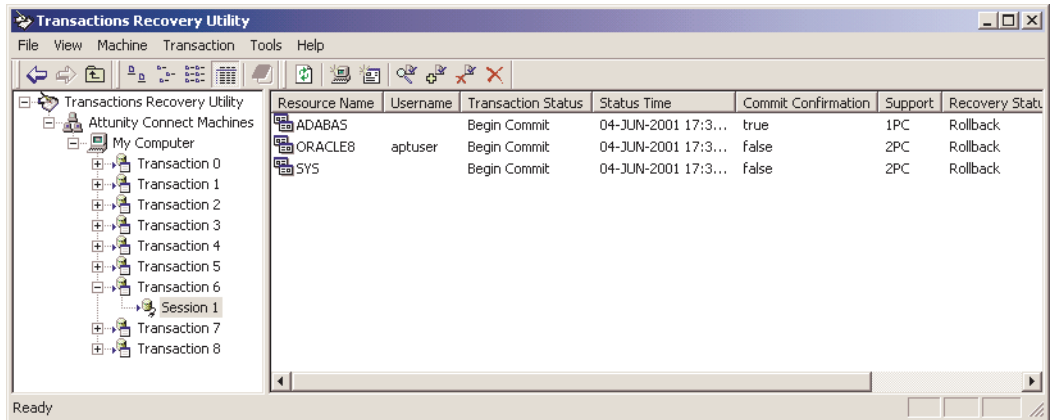
Generally, the ability to support distributed transactions depends on the capabilities of the data sources that participate in the transaction. Attunity Connect cannot guarantee data integrity for Attunity Connect-enabled data sources that do not support the 2PC protocol. However, Attunity Connect does employ various optimizations to extend the coverage of 2PC support. For example, Attunity Connect can execute a 2PC process as long it involves only one data source that does not support the 2PC protocol.

**Distributed Transactions Recovery Tool**

The Distributed Transactions Recovery tool provides a graphical environment in which administrators can browse transaction logs and perform manual or automatic recovery – for local or remote transaction logs. The transaction log file and recovery files serve as an aid to



recovery in the event of one- and two-phase commit failure on the part of the transaction coordinator.



Recovery can be manual or automatic. In a manual recovery, the user examines the transaction log status and decides the fate of each transaction. In a heuristic automatic recovery, the system decides the outcome of the transaction itself, based on the last state of each transaction.

Attunity Connect supports both client-initiated and server-initiated recovery. In client-initiated recovery, a system administrator uses a graphical interface to examine the status of all transactions before committing or rolling back the entire transaction. Server-initiated recovery ensures that a particular database is left in a consistent state and that no records remain locked due to 'in-doubt' transactions. In this case, the utilities recover (commit or roll back) all of the transactions in which this server participates, on this server only.

## Base Services

Attunity Connect includes a suite of administration and development tools for system and database administration. IT staff can run most of these utilities remotely using the GUI interface or locally on the native machine using the command line interface.

### Installation Wizards

Attunity Connect offers a native installation wizard on each of the supported platforms, simplifying deployment. It takes no special skills to install Attunity Connect, so IT staff can set up the Attunity Connect infrastructure by applying only their platform- and application-specific expertise.

**Attunity Connect  
Internal Storage (the  
Repository)**

Attunity Connect stores and accesses metadata (such as table definitions) and configuration items (such as user profiles) in repositories. There is a single internal repository for each Attunity Connect data source. There is also one system repository that maintains system-wide definitions and application adapter definitions. All repositories are optimized for fast run-time access and for maintaining definitions and are not restricted by native operating system file naming conventions.

**The system repository** – The system repository is used for information that is general to Attunity Connect, such as:

- Binding information.
- Daemon definitions.
- User profiles. (A user profile is a list of username/password pairs for machines and data sources. The user profile is used to access data, without having to be prompted for user information at every stage. Each user profile is accessible by a username and password pair.)
- Information used directly by Attunity Connect query processor (Attunity Connect procedures and views).

The system repository is called SYS.

**A data source repository** – A repository exists for each data source specified to Attunity Connect. The information in the repository includes:

- Metadata for the data source.
  - Attunity Connect metadata for non-relational data sources and for Attunity Connect procedures.
  - Attunity Connect metadata that is additional to the native metadata of the data source (the extended metadata feature) and a snapshot of native metadata (the local copy feature).
- Attunity Connect synonym definitions for the data.

**Attunity Studio:  
Configuration and  
Management GUI**

Administrators can use Attunity Studio during design time to configure:

- Binding configurations for data sources and application adapters.
- Client-server communication via the Attunity Connect daemon listener.
- User profiles, enabling access to data sources and applications at runtime, without being prompted for user names and passwords.
- Metadata for both data sources and adapters. For all relational data sources and some non-relational data sources,

Attunity Connect uses the native metadata. Otherwise, the metadata is specified to Attunity Connect.

Administrators can use Attunity Studio during runtime to manage Attunity Connect daemons and logging.

#### **Client/Server Communication and the Attunity Connect Daemon**

Attunity Connect takes advantage of daemons, client software, and server software to support seamless operations across both local and remote distributed environments.

#### **Daemons**

Attunity Connect includes daemons on all supported platforms. The daemons handle user authentication and authorization, connection allocation, and server process management. A fail-safe mechanism allows the specification of alternate daemons, in support of standby server machines with replicated data. System administrators can specify the desired detail of event logging and real-time usage information.

#### **Client Software**

Within Attunity Connect's symmetrical operation, clients serve as agents that request remotely located data. Client software, which includes one or more application-specific interfaces, resides on every system that needs to interact with data via Attunity Connect.

To the calling application, Attunity Connect clients look like local data providers. They receive requests for data and metadata and either execute those requests locally or dispatch them to an appropriate server. The communication protocol among Attunity Connect components minimizes processing and traffic by negotiating data formats (for example, when similar systems talk, there is no need to switch data formats) and by avoiding repeated transmission of the same data over the network.

The communication subsystem handles machine-dependent transformations such as big/little endian translations, floating point format translations, single- and multi-byte character encoding translations, etc. Attunity Connect dynamically determines these translations upon initial connection between a client and a server, based on the nature of the parties involved in the connection.

Clients maintain caches of data and metadata which enable it to satisfy many requests locally without needing to go to the server. They also batch some commands in order to avoid unnecessary network traffic.

Upon the first remote request by a particular user session, a client starts a corresponding session for this user on one or more servers.

Each server session remains open until the client session terminates. In the case of systems using connection pooling (such as MTS or IIS), the client and server sessions may stay open indefinitely. In the event that server operations terminate (for example, someone restarts the server machine or communication is lost), the client automatically reestablishes the connection upon the next remote operation.

### **Server Software**

Attunity Connect daemons support multiple server configurations. This capability gives organizations more control over the use of system resources and access rights for specific sources and applications. It also makes it easier to tune the system at the deployment site.

By isolating different servers, the configurations increase reliability and flexibility. They enable solutions to serve different clients, including classic two-tier client-server applications, three-tier applications with connection pooling, and ad-hoc usage.

Each server configuration specifies the server's:

- Processing mode, specifying multi-threaded/single-threaded operations, the number of processes, and server pools.
- Security settings for impersonation, authorized users, administrators, and encryption.
- Accessible adapters.
- Various operational parameters.

Attunity Connect also supports multi-version interoperability. IT teams can simultaneously install multiple versions of Attunity Connect on all supported operating systems. As a result, organizations can begin to deploy new software versions in a staged update process, without interrupting the operations of previous versions.

At the provider end of the system, Attunity Connect servers act as agents that access, read, manipulate, and write to data sources and applications. Attunity Connect servers accept commands from clients, call the corresponding local functions, and package and return the results to the clients.<sup>2</sup>

Server software, which includes the Attunity Connect engines, drivers and adapters, resides on every Attunity Connect machine.

---

<sup>2</sup> In fact, a target data source may actually reside on another machine. In this case, the source is represented by an agent on the server using a third-party communications component such as Oracle Net Servers or Sybase CT-Lib, which is transparent to Attunity Connect.

Whenever a client requests a new user session, the daemon allocates a server instance to the client. Thereafter, the client communicates with the server directly without the mediation of the daemon. This kind of server model is very flexible. It accommodates different operating systems and data source requirements.

Attunity Connect supports several server models:

- The multi-threaded model is effective when the data sources support multi-threading.
- Serialized multi-client server processes are useful for short requests and for data sources that allow more than one simultaneous client per process.
- The single-client-per-process model supports data sources that only handle one client per process and to maximize client isolation.

Attunity Connect can reuse server processes. Various server process pooling options allow organizations to tune the solution for different application and load requirements.

## Supported Platforms

Attunity Connect has been certified to run on the following platforms:

- HP NonStop Himalaya (NSK)
- HP OpenVMS on Alpha and VAX
- IBM OS/390 and z/OS
- IBM OS/400
- Red Hat Linux on Intel
- UNIX on Data General, HP (HP-UX and Tru64), IBM RS/6000 and Sun Solaris
- Microsoft Windows 98/NT/2000/2003/XP



## Chapter 2    **A Quick Walkthrough – Accessing Data**

The following tutorial provides a guided walkthrough to setting up and using Attunity Connect to access data.

Attunity Connect is configured in Attunity Studio.

The following steps of setting Attunity Connect to access to data are explained in this tutorial:

- Setting up access to data on a client machine
  - Setting access to the machine
  - Setting up access to data on the machine
- Setting access to data on a server machine
  - Setting access to the machine
  - Setting up access to data on the machine
- Connecting to multiple data sources

### **Setting Up Access to a Client Machine to Access Data**

Setting up access to data using Attunity Connect requires setting up access to the machine where the data resides and setting up access to the data source.

When configuring Attunity Connect, you start by adding the machine running Attunity Connect in Attunity Studio.

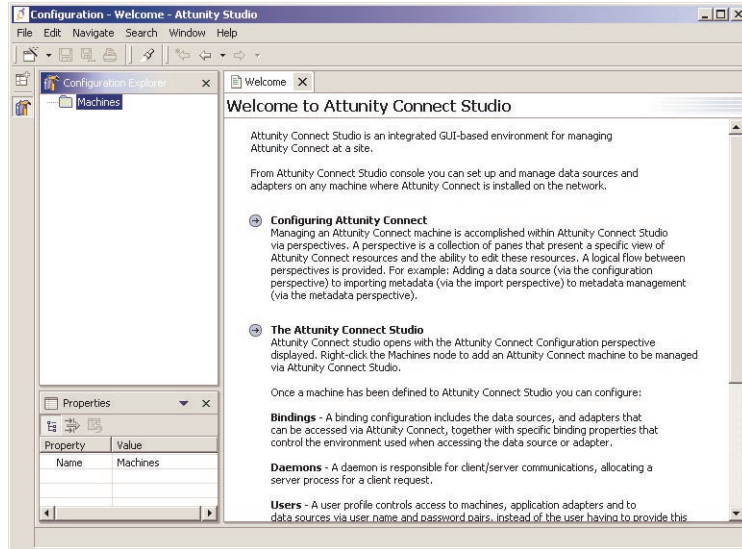
### **Setting Up Access to a Machine**

- ❖ The following procedures assumes:
  - You have permission to access the server machine.
  - The Attunity Connect daemon is running on every machine used in this tutorial.

Check with the system administrator to ensure these requirements are fulfilled.

► **To add a machine:**

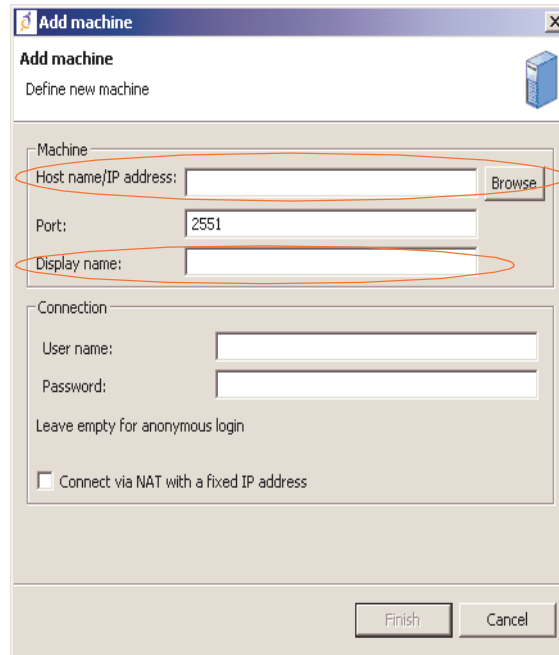
1. **Open Attunity Studio by selecting Start | Programs | Attunity | Studio1.2.**



2. **Right-click Machines in the Configuration explorer and select Add Machine.**

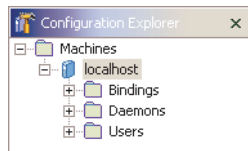


The Add Machine window is displayed:



3. Click the **Browse** button and select the client machine (the machine you are working on) from the list of machines which have Attunity Connect running on the displayed port.
  - ❖ The client machine set here is the machine where the application runs.
4. Optionally, enter an alias for the machine in the Display name field. For example, you can change the name to localhost.
5. Leave the Username and Password fields empty to connect anonymously.
  - ❖ On Windows machines, Anonymous connection is the default.
6. Click **Finish**.

The machine now appears in the Configuration Explorer tree.



Every Attunity Connect machine has the following areas that can be configured:

**Binding configuration** – The binding configuration lists the data sources and application adapters that reside on the machine.

**Daemon configuration** – The daemon configuration controls client server communication.

**User configuration** – The user configuration manages runtime authorization rights to access data sources, application adapters and remote machines from the client machine.

## Setting Up Access to Data

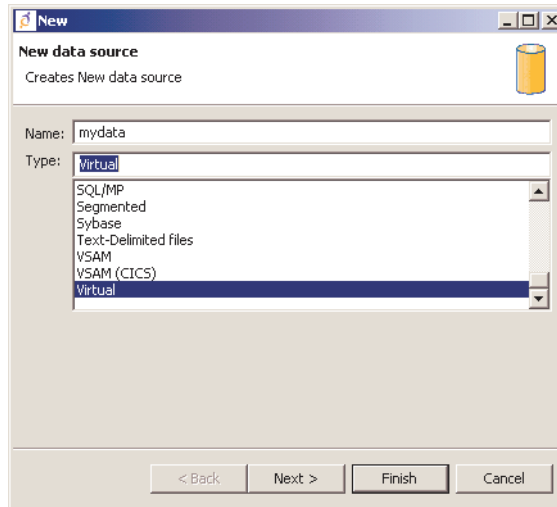
To access a data source, you need to define the data source in the binding configuration.

NAV is the default binding configuration for Attunity Connect. You can use this configuration to define all the data sources and adapters you want to access via Attunity Connect. You can also create a new binding configuration.

### ► Setting up access to a data source:

1. Create a new directory on the server machine where the data source resides. This directory will be used to store new data that you will create as part of this tutorial. The directory used in this example is called *mydata*.
  - ❖ Although the process described here is valid for all supported platforms, for simplicity, the Windows platform is used in the example.
2. Expand Bindings under the machine in the explorer tree.
3. Expand the NAV binding. The binding contains branches for data sources, adapters and events.
4. Right-click Data sources and select New Data source.

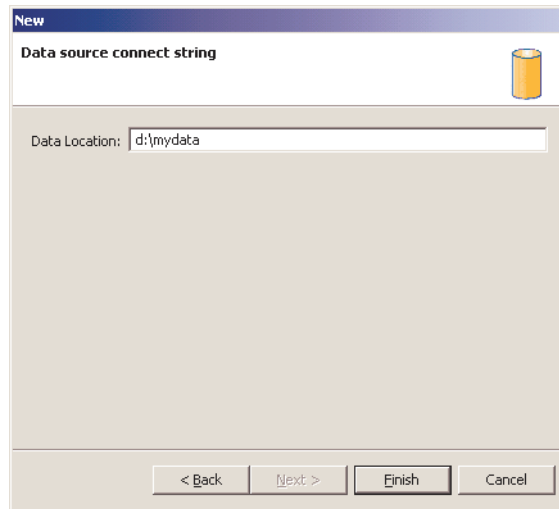
5. In the New data source wizard, enter a name for the data source and select Virtual as the data source type.



Virtual is an Attunity Connect virtual data source. You can use this data source on any machine, regardless of what data sources are registered for use by Attunity Connect.

6. Click **Next**.

7. Enter connection details for the data source. Enter the full path where files and indexes you create with CREATE TABLE statements are stored.
  - ❖ The directory specified (in the example d:\mydata) **must** exist on the machine in order to create tables for the data source.



8. Click **Finish**.

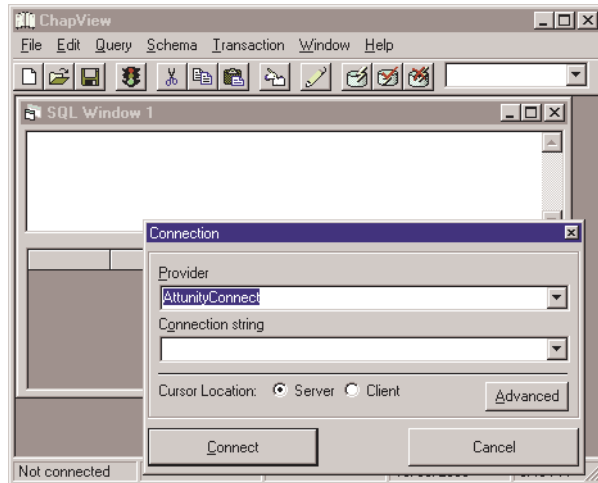
The data source is now displayed in the Configuration explorer tree.

## Testing the Connection

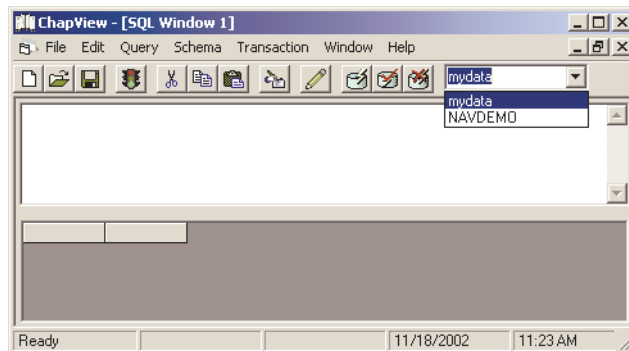
You can ping the data source in Attunity Studio, to ensure that the data source can be accessed. The following test includes creating a table for the data source, using the Attunity Connect demo ADO application.

- **To connect to the data using the demo ADO application:**
  1. Start the demo ADO application by selecting **Start | Programs | Attunity | Demos | Demo ADO Application**.

The Attunity Connect demo ADO application opens displaying a Connection screen:



2. Click **Connect** to connect to Attunity Connect as the data provider.
3. Select the mydata as the data source you want to test.



4. In the SQL Window, enter the following SQL:

```
create table tutorial (id integer, lastname
varchar(20) not null, firstname varchar(20) not
null)
```

- ❖ No message is returned stating that the statement was executed successfully. If you try to execute the statement a second time, you receive an error (since the table already exists).
- ❖ The CREATE TABLE statement also creates the necessary metadata for the table in Attunity Connect.

5. Click  to execute the SQL.

6. Enter a row to the table. For example:

```
insert into tutorial values(1, 'white', 'julian')
```

Click  to execute the statement.

❖ When the execution is successful no message is displayed.

7. Enter more rows to the table. For example:

```
insert into tutorial values(2, 'black', 'stuart')
```

```
insert into tutorial values(3, 'brown', 'alan')
```

Click  after each insert statement.

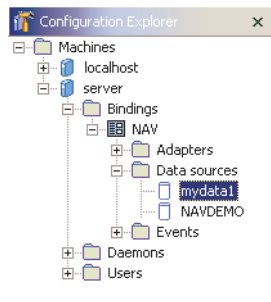
8. Use a SELECT statement to check that the data was entered correctly:

```
select * from mydata:tutorial
```

9. Click  to execute the statement.

## Setting Up Access to Data on a Server Machine

1. Add the server machine as you added the client machine in "Setting Up Access to a Machine", at the beginning of the tutorial.
  - ❖ The client machine is where the application runs and the server machine is where the data source resides. Every machine is added in the same way to Attunity Studio, except that non-Windows machines might require a username and password, depending on whether this was specified during the installation (the default is that anyone can initially administer a machine from Attunity Studio).
2. Add a data source, called mydata1, to the server machine as you did in "Setting Up Access to Data", above.



Once the server machine and data on it are set, you can access it from the client machine.

► **To access data on a server machine via the client machine:**

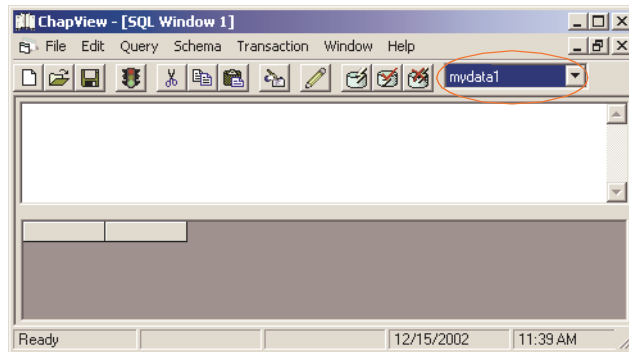
1. Select the mydata1 data source and drag-and-drop it to the Data sources node on the client machine.
2. Choose Create Shortcut from the popup menu.

The data source shortcut is displayed in the Configuration explorer under client machine.

## Testing the Data Source Shortcut

► **To connect to the data source on the server machine:**

1. In the demo ADO application, select mydata1 as the data source you want to test.



2. In the SQL Window, enter an SQL CREATE TABLE statement to create a new table to the data source you specified as mydata1, as follows:

```
create table tutorial1 (id integer, lastname
varchar(20) not null, firstname varchar(20) not null)
```

- ❖ No message is displayed if the execution is successful.
- ❖ The CREATE TABLE statement also creates the necessary metadata in Attunity Connect for the table.

3. Click  to execute the SQL.

4. Enter rows to the table. For example:

```
insert into tutorial1 values(1, 'brown', 'betty')
insert into tutorial1 values(2, 'smith', 'steve')
insert into tutorial1 values(3, 'williams', 'bill')
```

- ❖ No message is displayed if the execution is successful.

Click  after each select statement.

5. Use a SELECT statement to check that the data was entered correctly. For example:

```
select * from mydata1:tutorial1
```

6. Click  to execute the statement.

## Connecting to Multiple Data Sources in a Single SQL Statement

To connect to multiple data sources in a single SQL statement, identify the data sources in the SQL statement by using the names assigned to them in Attunity Connect. In this example mydata and mydata1. Separate the data source name and the table name with a colon (:).

The following example joins the tutorial table from mydata to mydata1 in tutorial1. The id field value is taken from the tutorial table and the lastname from the tutorial1 table.

```
select tutorial.id, tutorial1.lastname  
  from mydata:tutorial,  
       mydata1:tutorial1
```



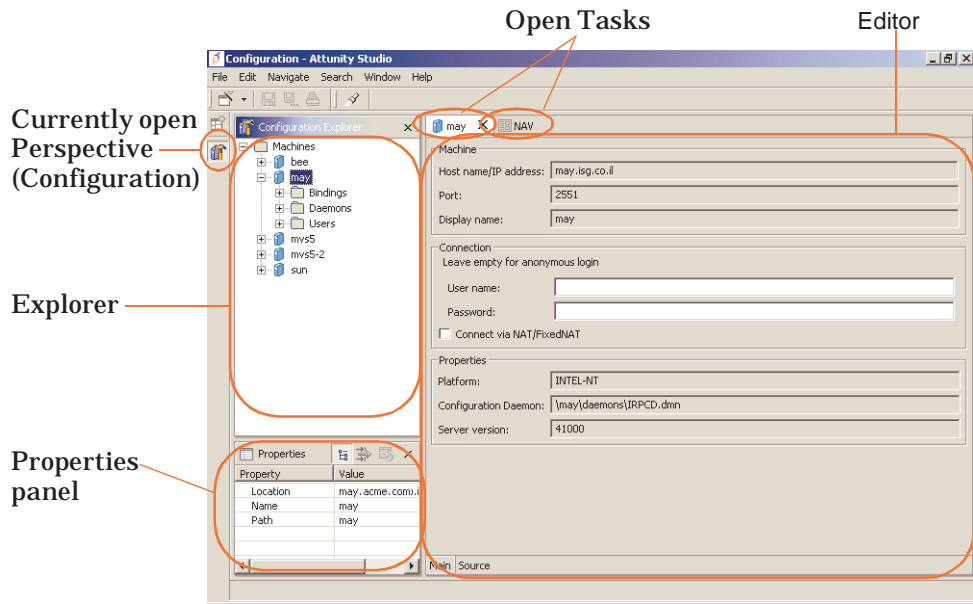
# Chapter 3    Setting Up Access to Data and Applications

Attunity Studio is used to configure and manage access to data and applications on all machines running Attunity Connect.

The following perspectives are available:


- **Configuration** – Used to configure access to machines, data sources and adapters.
- **Metadata** – Used for the following:
  - To manage Attunity Connect metadata (ADD) for data sources that either don't have metadata (such as DISAM), or have metadata that cannot be used by Attunity Connect.
  - To manage Application adapter definitions.
  - To extend metadata in ADD, for relational data sources that require additional information not supplied by the native metadata (such as some statistics for some relational data sources).
  - To manage a snapshot of relational metadata converted to ADD (also referred to as local copy metadata).
  - To view relational metadata.
- **Metadata Import** – Used for the following:
  - To import metadata for data sources that require ADD.
  - To import adapter definitions for application adapters.
- **Runtime Manager** – Enables monitoring Attunity Connect daemons.

The following is an example of Attunity Studio with the Configuration perspective open:



A perspective consists of an explorer area and an editor area. The explorer is used to navigate and manage the resources. The editor area is used to perform the main tasks. The Properties panel displays resource properties such as the location of the selected data source or adapter.

- ❖ The Properties panel is useful to identify on what machine a data source or adapter is located, especially when several data sources or adapters on different machines have similar names. Identifying the location is particularly useful in the Metadata and Metadata Import perspectives.

**Opening a Perspective** Open a perspective by clicking the Open a Perspective button  and selecting a perspective from the list, or via **Window | Open Perspective**. Attunity Studio opens with the Configuration perspective displayed.

## Configuring Access to Machines with Data or Applications

The Attunity Studio Configuration perspective enables configuring access to Attunity Connect machines and to data and applications on those machines. Access to a machine is configured by defining the name, port and login information for the machine.

► **To add a machine to Attunity Studio:**

1. Open Attunity Studio by selecting **Start | Programs | Attunity | Studio1.2**.
2. In the Configuration explorer, right-click Machines in the Configuration explorer, and define the machine as described below:

**Add machine**  
Define new machine

**Machine**  
Host name/IP address:  Browse  
Port:   
Display name:

**Connection**  
User name:   
Password:   
Leave empty for anonymous login  
☐ Connect via NAT with a fixed IP address

Finish Cancel

Access to the machine is set by specifying the following information.

**Host name/IP address** – The name of the machine on the network. The name can be entered manually or found using the Browse button, which lists all the machines running an Attunity Connect daemon listener on the specified port currently accessible over the network.

**Port** – Specifies the port where the Attunity Connect daemon is running. The default port for Attunity Connect is 2551.

**Display name** – (Optional) An alias used to identify the machine when different from the host name.

**User name** – (Optional) The username of a user defined as an administrator for the machine.

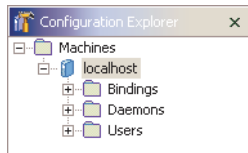
- ❖ An administrator is specified when the machine is installed or using NAV\_UTIL ADD\_ADMIN on the machine itself, as described in *Attunity Connect Reference*.

**Password – (Optional)** The password of the user.

**Connect via NAT with a fixed IP address** – Specifies whether the machine uses the NAT firewall protocol.

After the machine has been added, it is displayed in the Configuration Explorer. Under each machine are the following entries:

- Bindings
- Daemons
- Users



## Bindings

The binding configuration lists the application adapters, data sources and events that reside on the machine and shortcuts to data sources that reside on other machines, and which are accessible from the specified machine.

- ❖ The configuration supplied with Attunity Connect includes the default NAV binding. This binding configuration is used if a binding is not specified when accessing a data source or application adapter.

You can have a number of binding configurations, each with a set of application adapters, data sources and events that can be accessed. Each configuration may have its own environment that defines the binding (such as cache sizes for storing information in memory during a session).

## Daemons

An Attunity Connect daemon runs on every machine. The daemon is responsible for allocating an Attunity Connect server process for a client.

The daemon authenticates clients, authorizes requests for a server process within a certain server workspace and provides the clients with the required servers. When a client requests a connection, the daemon allocates a server process to handle this connection, and refers the client to the allocated process.

**Users**

The user configuration sets the runtime authorization rights to access data sources, application adapters and server machines from the client machine.

The Users List includes the default NAV user profile.

**Using an Offline Design Machine to Create Attunity Connect Definitions**

Offline design mode enables you to define the resources to Attunity Connect within Attunity Studio, without having to connect to the actual server machine where these definitions will be implemented. Thus, for example, you can set up a machine even when the actual server machine is down, or set up a number of definitions for different machines on the same design machine.

Once the resources have been defined on the design machine, you implement the definitions by dragging-and-dropping each definition to the specific server machine where you want the definition implemented.

► **To define an offline design machine:**

1. Right-click Machines in the Configuration explorer.
2. Select Add Offline Design Machine from the popup menu.
3. Enter a name for the Design Machine.
4. Click **Finish**.

You can define all available resources on this machine. You can also set up metadata using a metadata import utility. Every resource is available on the design machine, irrelevant of the actual platform where the resource needs to exist. Thus, for example, both HP NonStop data sources and OS/390 data sources are available, even though on completion, you can drag-and-drop the NonStop definitions (such as an Enscribe data source) to an HP NonStop machine.

**Creating a New Binding Configuration**

Each binding configuration includes the following:

- Environment settings
- Machines where data source that are to be accessed from the current machine reside.
- Data sources on the current machine
- Shortcuts to data running on other machines
- Application adapters on the current machine
- Events on the current machine

You can set up a number of different configurations, each for different requirements of the same application adapters, data sources or events. For example, you can set up two different configurations allowing different users access to specific resources.

In Attunity Studio, bindings are defined in the Configuration perspective.

► **To add a new binding configuration:**

1. Open Attunity Studio by selecting **Start | Programs | Attunity | Studio1.2**.
2. In the Configuration explorer, open the machine you want to set the binding for.
  - ❖ You can add a new binding configuration in offline design mode, in a design machine and later drag-and-drop the binding to this machine, as described in "Using an Offline Design Machine to Create Attunity Connect Definitions" on page 37.
3. Right-click Bindings and select New Binding from the popup menu.
4. Enter a name for the binding in the New Binding window.
5. Click **Finish**. The new binding editor, used to set the new binding, opens, displaying the properties for the binding. See "Setting the Binding Environment" below for details of the binding properties.

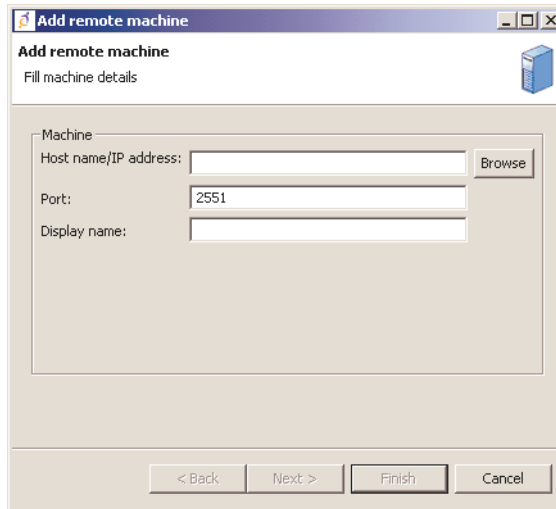


► **To set up access to another machine**

- [illegible]



2. Use the Add remote Machines window to define the machine access information.



3. Click **OK**.
  - ❖ If the users allowed access to the other machines are known at this point, the user profile for the machine can be set by clicking the Security button in the Machines tab. Otherwise, the users are set from the users node of the Configuration Explorer, as described in "User Profile Security" on page 96.

## Adding a Data Source

A data source is defined on the machine where it resides. Data sources residing on other machines and accessed by the current machine are set as a data source shortcuts. See "Adding a Data Source Shortcut" on page 42 for details.

### ► To add a data source to a binding configuration:

1. Open Attunity Studio by selecting **Start | Programs | Attunity | Studio1.2**.
2. In the Configuration perspective, expand the machine you want to add the data source to, in the explorer tree.
  - ❖ You can add the data source in offline design mode, in a design machine and later drag-and-drop the data source to this machine, as described in "Using an Offline Design Machine to Create Attunity Connect Definitions" on page 37.
3. Expand Bindings and expand the binding configuration, to which you want to add the data source.

4. Right-click Data sources and select New Data source.
5. Enter a name for the data source.
  - ❖ The name must be unique to the binding configuration.
6. Select the type of the data you want to access from the list.
7. Click **Next**.
8. Enter a connect string for the data source you are connecting.

For the connect string format appropriate for the data source you are accessing refer to the specific data source in *Attunity Connect Reference*.
9. Click **Finish**.

The data source is displayed in the Configuration explorer tree. An orange database symbol represents a data source that requires Attunity Connect metadata. A table symbol over an orange database symbol represents a data source that does not require Attunity Connect metadata. A white database symbol represents a data source that uses the Attunity Connect Virtual driver.

If the data source requires specific configuration properties, these are entered in the Properties tab of the data source editor. Details of properties for specific data source are provided in *Attunity Connect Reference*.

## Adding a Data Source Shortcut

A data source shortcut is a link on the client machine to a data source on another machine.

► **To add a data source shortcut to a binding configuration by dragging-and-dropping:**

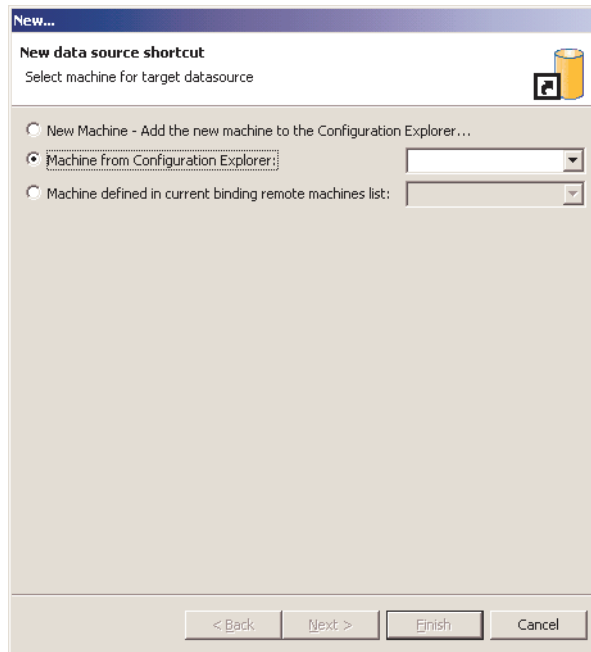
1. Select the data source on the server machine and drag-and-drop it to the Data sources node under the binding on the client machine.
2. Choose Create Shortcut from the popup menu.

The data source shortcut is displayed in the Configuration explorer under client machine.

► **To add a data source shortcut to a binding configuration using the wizard:**

1. Open Attunity Studio by selecting **Start | Programs | Attunity | Studio1.2**.
2. In the Configuration perspective, expand the machine you want to add the data source shortcut to, in the explorer tree.

- ❖ You can add the data source shortcut in offline design mode, in a design machine and later drag-and-drop the shortcut to this machine, as described in "Using an Offline Design Machine to Create Attunity Connect Definitions" on page 37.
3. Expand Bindings and the binding configuration to which you want to add the data source shortcut.
  4. Right-click Data Sources and select New Data source Shortcut. The following window is displayed.



5. In the New data source shortcut window specify if the machine you want to set the shortcut to is:
  - **New Machine** – not defined in Attunity Studio.
  - **Machine from Configuration Explorer** – defined in the Configuration perspective.
  - **Machine defined in current binding remote machine list** – defined as a remote machine in the same binding as the shortcut is being added to. For details see "Setting Up Access to Other Machines" on page 40.
6. If you selected New in the previous step, in the Add Machine window set the machine as in "Setting Up Access to Other Machines" on page 40.



## Adding an Adapter

► **To add an adapter to a binding configuration:**

1. Open Attunity Studio by selecting **Start | Programs | Attunity | Studio1.2**.
2. In the Configuration perspective, expand the machine you want to add the adapter to, in the explorer tree.
  - ❖ You can add the adapter in offline design mode, in a design machine and later drag-and-drop the adapter to this machine, as described in "Using an Offline Design Machine to Create Attunity Connect Definitions" on page 37.
3. Expand Bindings and expand the binding configuration, to which you want to add the adapter.
4. Right-click Adapters and select New Adapter.
5. Enter a name for the adapter.
  - ❖ The name must be unique to the binding configuration.
6. Select the adapter type from the drop-down list of possible adapters.
7. Click **Finish**.
  - ❖ If the Attunity Studio preferences have been set to enable the same adapter definition to be used for multiple adapters, you are prompted to specify which adapter definition describing an adapter from the same binding to use

The adapter is displayed in the Configuration Explorer tree.

If the adapter requires specific configuration properties, these are entered in the Properties tab of the Adapter Editor. Details of properties for specific adapters are provided in *Attunity Connect Reference*.

The Properties tab of the Adapter Editor is accessed by right-clicking the adapter and selecting the Edit Adapter option and opening the Properties tab.

## Adding an Event

An event is specified case of an adapter. When you define an adapter you can also specify that you want to define an event which is linked to the adapter.

► **To add an event to a binding configuration:**

1. Open Attunity Studio by selecting **Start | Programs | Attunity | Studio1.2.**
2. In the Configuration perspective, expand the machine you want to add the event to, in the explorer tree.
  - ❖ You can add the event in offline design mode, in a design machine and later drag-and-drop the event to this machine, as described in "Using an Offline Design Machine to Create Attunity Connect Definitions" on page 37.
3. Expand Bindings and open the binding configuration, to which you want to add the event.
4. Right-click Events and select New Event.
5. Enter a name for the event.
  - ❖ The name must be unique to the binding configuration.
6. Click **Finish**.

The event is displayed in the Configuration explorer tree.

## Chapter 4    **Configuring Client/Server Communication**

An Attunity Connect daemon runs on every machine running Attunity Connect. The daemon is responsible for allocating server processes to clients.

The daemon authenticates clients, authorizes requests for a server process within a certain server workspace and provides clients with the required servers. When a client requests a connection, the daemon allocates a server process (or where applicable, a thread) to handle this connection, and refers the client to the allocated process. This may be a new process (dedicated to this client) or an already-started process. Further communication between the client session and the server process is direct and does not involve the daemon. The daemon is notified when the connection ends and the process is either killed or remains available for use by another client

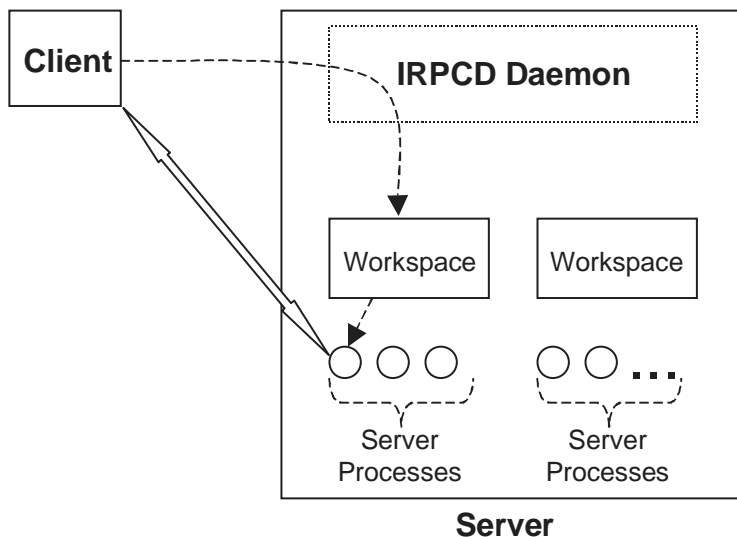
- ❖ If a client session ends due to a power cut, network inaccessibility or a system reset, the associated server process may remain for a long time and may have to be killed by the system administrator.  
You can monitor the status of all the currently active Attunity Connect server processes to identify server processes that may have to be killed. For details refer to *Attunity Connect Reference*.

### **OS/390 and z/OS Platforms**

The Attunity Connect daemon resides in a single address space. The Attunity Connect server process is executed as a started task.

The daemon supports multiple server configurations called *workspaces*. Each workspace defines accessible data sources and adapters, environment settings, security requirements and server allocation rules. The allocation of servers by the daemon is based on the workspace that the client uses to access the data source. Thus, for example, a client can access a data source via one workspace, where a server process is allocated from an existing pool of servers, or the client

can access a data source via a different workspace, where a new server process is allocated for each client request.



- ❖ The maximum number of concurrent server processes per workspace and/or of concurrent open connections is a configuration parameter.

## Starting the Daemon

You start a daemon from a privileged account (such as the super user account on a UNIX platform) on the machine where the daemon will run. If not run from a privileged account, the daemon can start servers only with the same user ID as the account that started it. In this case, the daemon may also have problems validating user name/password pairs within the system.

The way you start the daemon depends on the platform.

- ❖ For special cases and additional information, refer to *Attunity Connect Reference*.



► **To start the daemon:**

- Enter the following at the command line of the server:

```
irpcd start.
```

#### **Windows Platforms**

Run this command through the Attunity Connect Environment Prompt menu item in the Attunity Connect menu (**Start | Programs | Attunity | Connect | Attunity Connect Environment Prompt**).

In the Control Panel Services change the service for the daemon from the System account to an account of your choice.

#### **OS/400 Platforms**

```
Sbmjob cmd(call pgm(navroot/irpcd start))
```

#### **OpenVMS Platforms**

The daemon should start automatically when the system boots up, through SYSSSTARTUP:NAV\_START.COM (see "Automatic Startup" in the *Attunity Connect Installation Guide* for more details). When IRPCD initializes itself as a daemon, it creates a detached process under the same account from which 'IRPCD start' was issued. In the detached process, the account's login procedure is not executed. If the daemon fails to start in the detached process, define the symbol NV\_DEBUG\_MODE to something before starting the daemon. This creates a process log file in SYSSLOGIN:IRPCD\_START.LOG which can help you to locate the problem.

#### **OS/390 and z/OS Platforms**

1. Ensure the following:

- The 'NAVROOT.loadaut' library is APF authorized.  
NAVROOT is the high level qualifier specified during installation.
- ❖ To define a DSN as APF authorized, in the SDSF screen enter the following command:  
"/setprog apf,add,dsn=navroot.loadaut,volume=nav002"  
where nav002 is the volume where you installed Attunity Connect.
- 'NAVROOT.USERLIB(ATTSRVR)' and 'NAVROOT.USERLIB(ATTTAEMN)' have been copied to a library within the started tasks path. If they have not been copied, add the 'NAVROOT.USERLIB' library to this path.

2. Activate 'NAVROOT.USERLIB(ATTDAEMN)' as a started task to invoke the daemon. For example, in the SDSF screen enter the following:

```
' /s ATTDAEMN'
```


To submit the daemon as a job, uncomment the first two lines of the ATTDAEMN JCL and run the job using the sub command. The ATTDAEMN JCL is similar to the following:

```
//*ATTDAEMN JOB
'RR', 'TTT', MSGLEVEL=(1,1), CLASS=A,
//*MSGCLASS=A, NOTIFY=&SYSUID, REGION=8M
//STEP1 EXEC PGM=IRPCD,
// PARM='-B START ' 'NAVROOT.DEF.IRPCDINI' ' '
//STEPLIB DD DSN=NAVROOT.LOADAUT, DISP=SHR
//SYSPRINT DD SYSOUT=A
//GBLPARMS DD DSN=NAVROOT.DEF.GBLPARMS, DISP=SHR
// EXEC
PGM=IRPCD, COND=((1,EQ,STEP1), (2,EQ,STEP1)),
// PARM='-KATTDAEMN START
' 'NAVROOT.DEF.IRPCDINI' ' '
//STEPLIB DD DSN=NAVROOT.LOADAUT, DISP=SHR
//SYSPRINT DD SYSOUT=A
//GBLPARMS DD DSN=NAVROOT.DEF.GBLPARMS, DISP=SHR
//SYSDUMP DD DUMMY
```

- ❖ You can also run ATTDAEMN by submitting the job, without making any changes to the JCL.

## Runtime Manager Perspective

The Runtime perspective enables monitoring daemon activity on any machine running a daemon.

Open the Runtime perspective by right-clicking a machine in the Configuration perspective and choosing **Open Runtime Perspective**, or by clicking the **Open a Perspective** button  and choosing **Runtime Manager**, or via the **Window | Open Perspective** menu.

To add daemons you want to monitor, right-click the **Daemons** node and choose **Add Daemon** from the popup menu.

The following window is displayed:

**Add daemon**  
Add new daemon

**Machine**  
Host name/IP address:  Browse  
Port:   
Display name:

**Connection**  
Leave empty for anonymous login  
User name:   
Password:   
☐ Connect via NAT with a fixed IP address

Finish Cancel

Access to the machine is set by specifying the following information.

**Host name/IP address** – The name of the machine on the network. The name can be entered manually or found using the Browse button, which lists all the machines running an Attunity Connect daemon listener on the specified port currently accessible over the network.

**Port** – Specifies the port where the Attunity Connect daemon is running. The default port for Attunity Connect is 2551.

**Display name** – (Optional) An alias used to identify the machine when different from the host name.

**User name** – (Optional) The username of a user defined as an administrator for the machine.

- ❖ An administrator is specified when the machine is installed or using NAV\_UTIL ADD\_ADMIN on the machine itself, as described in *Attunity Connect Reference*.

**Password** – (Optional) The password of the user.

**Connect via NAT with a fixed IP address** – Specifies whether the machine uses the NAT firewall protocol. Once a machine is added you can drill down to its workspaces and the server processes for each workspace.

**Managing a Daemon**

The following options are used to manage the daemon, workspace and server settings. These options are accessed by right-clicking a resource:

**Status** – Checks the status of the daemon, workspace or server.

**Reload Configuration** (at daemon level) – Reloads the configuration after any changes. Any started servers are not effected by the changed configuration.

❖ The configuration must be reloaded to apply changes.

**View Log** – View the log for the selected daemon, workspace or server. The details displayed are the same as those written to the daemon log or the server log only that they are displayed only from the time the view is relevant. The number of servers the can be managed depend on the monitor section of the daemon control.

**View Events** – View the activity on the daemon, workspace or server selected. The event router provides a filtered representation of the log monitor.

**Daemon Properties** (at daemon level) – Information about the machine the on which the daemon is running.

**Shutdown Daemon** (at daemon level) – Shuts down the daemon on the machine.

❖ You can only start a daemon from the machine command line and not from the Attunity Studio.

**End Unused Servers** (at daemon and workspace levels) – Refreshes the daemon either for all workspaces (the daemon level) or for the specified workspace. Changing various server properties affects only new server processes. All available and unconnected servers are terminated and any connected servers are marked and terminated on release. Use this option when changes to the binding configuration or to the environment settings were made after servers were started up. On the next operation, servers are restarted, based on the available server and reusable server settings.

❖ This option has no meaning unless the Keep When Daemon Ends field is checked in the WS Info tab for the Workspace.

**End All Servers** (at daemon and workspace levels) – Kills all the active servers for all workspaces (the daemon level) or for the specified workspace, regardless of whether the server has an active client.

**End** (at server level) – Kills the server.

**Rename** (at daemon level) – Rename the daemon.

**Remove** (at daemon and workspace levels) – Removes the daemon or workspace with its servers from the Runtime explorer. If the workspace was removed, you can add it back by right-clicking the daemon and choosing Refresh from the popup menu.

**Edit Daemon Configuration** (at daemon level) – Opens the Daemon editor to update the daemon configuration.

**Edit Workspace Configuration** (at workspace level) – Opens the Workspace editor to update the daemon configuration.

**Refresh** – Refreshes the display.

**Import XML definitions** (at daemon level) – Imports daemon definitions from an XML file.

**Export XML definitions** (at daemon level) – Exports the daemon definition to an XML file. Before you export the definition, it is displayed.

#### **Logs in the Runtime Perspective**

Attunity Studio manages the following logs that you can use to troubleshoot problems:

**Daemon log** – Displays activity between clients and the daemon, including clients logging in and logging out from the daemon.

**Workspace log** – Displays information about the workspace being used by the client.

**Server process log** – Displays activity between clients and the server process used by that client to handle the client request.

The Runtime perspective of Attunity Studio provides a monitor for these logs.

Display the required log by right-clicking the level you want (daemon, workspace or server) and choosing the View Log option.

#### **Determining What is Displayed in the Log**

You can change the level of logging by clicking the Properties button. The following levels of logging are available:

**none** – The log displays who has logged in and out from the daemon.

**error** – The log displays who has logged in and out from the daemon and any errors that have been generated.

**debug** – The log displays who has logged in and out from the daemon, any errors that have been generated and any tracing that has been specified in the daemon configuration.

## Shutting Down the Daemon

### ► To shut down the daemon using Attunity Studio:

The daemon is shut down using Attunity Studio, in the Runtime perspective.

1. In the Runtime perspective, right-click the daemon you want to shut down.
2. Choose Shutdown Daemon in the popup menu.

### ► To shut down the daemon on non-windows platforms

- Run the following command in the command line:.

#### **HP (Compaq) NonStop, OpenVMS, UNIX and Windows Platforms**

```
irpcd shutdown
```

You can also issue `irpcd -s stop`.

#### **OS/390 and z/OS Platforms**

```
NAVROOT.USERLIB(IRPCDCMD)
```

Enter “shutdown” at the prompt or enter a control command:

```
/P ATTDAEMN or /F ATTDAEMN, STOP
```

#### **OS/400 Platforms**

```
call pgm (navroot/irpcd) parm(shutdown)
```

#### **HP (Compaq) NonStop, OpenVMS, OS/400, UNIX and Windows Platforms**

**oper** – Shuts down the daemon by sending a signal (SIGQUIT) to the daemon process. You do not need to specify the username/password for this option but you must have system privileges (you need to be a superuser). To send a signal to the daemon, the IRPCD program requires the process ID of the target daemon: the program retrieves this information from the file `irpcd[_port].pid` in the directory where the daemon resides (a daemon that was started on a particular port would have the port number in the PID filename). The PID file is automatically created when a daemon starts and is deleted when a daemon ends.

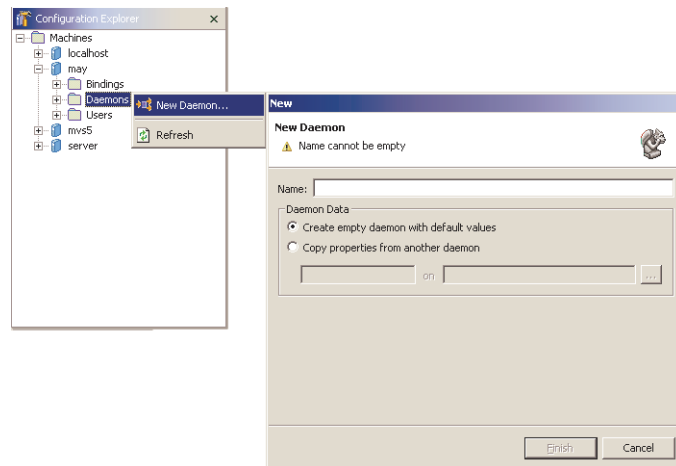
## Configuring a Daemon

Use Attunity Studio to maintain daemon parameters for all daemons on all machines. The daemon can be initially configured from the Configuration perspective. Normally, you will make changes to the daemon configuration after monitoring it in the Runtime perspective. In this case, you access the daemon configuration by selecting a daemon, right-clicking and choosing either Edit Daemon Configuration or Edit Workspace Configuration from the popup menu.

You can also have a number of daemon configurations for the machine.

► **To add a new daemon configuration:**

1. Open Attunity Studio in the Attunity Connect menu (**Start | Programs | Attunity | Studio1.2**).
2. In the Configuration explorer expand the machine to which you want to add the daemon. Right-click Daemons and select New Daemon from the popup menu.
  - ❖ A machine can have a number of daemons running, at the same time, each on its own port.



- ❖ You can add a new daemon configuration in offline design mode, in a design machine and later drag-and-drop the daemon configuration to this machine, as described in "Using an Offline Design Machine to Create Attunity Connect Definitions" on page 37.
3. In the New Daemon window, specify a name the new daemon and specify whether you want it to have default settings or to copy the properties of an existing daemon.

To copy the properties of an existing daemon, click the ellipsis (...) button and select the daemon from which you want to copy the properties.

Once the daemon is set the Daemon editor is displayed.

## Editing the Daemon

The daemon editor is divided into tabs, focusing on general settings, daemon logging and daemon security.

- You configure the daemon using the following tabs:
  - **Daemon Control** – Specifies the server details, including daemon failure recovery, maximum request file size, default language, and time out parameters.
  - **Daemon Logging** – Specifies the logging details such as, the logfile format and location, and the parameters to log and trace.
  - **Daemon Security** – Specifies administrative privileges and machine access are set in the tab.

Workspaces are added by right-clicking the daemon in the Configuration Explorer and selecting Add Workspace.

- You can edit a workspace using the following tabs:
  - **WS Info** – Specifies general information including the server type, the command procedure used to start the workspace, the binding configuration associated with this workspace and the timeout parameters.
  - **WS Server Mode** – Specifies workspace server information including features that control the operation of the servers started up by the workspace and allocated to clients.
  - **WS Logging** – Specifies parameters to follow and log and the format to use for the log file.
  - **WS Security** – Specifies administration privileges, user access, ports available for access to the workspace and workspace account specifications.
  - **WS Governing** – Specifies the way queries are to be executed.

### Configuring the Server Mode

The server mode dictates how the daemon starts up new server processes. The daemon supports the following server modes:

**Single Client** – Each client receives a dedicated server process. The account in which a server process runs is determined either by the client login information or by the specific server workspace.

This mode enables servers to run under a particular user account and isolates clients from each other (since each receives its own process).



However, this server mode incurs a high overhead due to process startup times and may use a lot of server resources (since it requires as many server processes as concurrent clients).

**Multi-Client** – Clients share a server process and are processed serially.

This mode has low overhead since the server processes are already initialized. However, because clients share the same process, they may impact one another, especially if they issue lengthy queries.

- ❖ When accessing a database that supports two-phase commit through XA, do not specify this mode.
- ❖ When accessing Adabas or DBMS, do not specify this mode.

The number of clients that share a process is determined by the “Clients per server limit” field (the maximum number of concurrent clients a server process for the current workspace accepts) in the Attunity Studio Configuration perspective.

- ❖ This value is set in the daemon configuration settings via the `maxNCLientsPerServer` parameter using the text editor. In Attunity Studio, this parameter is set in the WS Info tab of the daemon workspace editor.

### Windows Platforms

**Multi-Threaded** – Clients are allocated a dedicated thread in a shared server process.

This mode has low overhead since the servers are already initialized. However, because clients share the same process, they may impact one another, especially if the underlying database is not multi-threaded.

- ❖ Multiple multi-client and multi-threaded servers may be started simultaneously for optimal performance.

When accessing a database that supports two-phase commit through XA, do not specify this mode.

The number of multi-threaded clients that share a process is determined by the “Clients per server limit” field (the maximum number of concurrent clients a server process for the current workspace accepts) in the Attunity Studio Configuration perspective.

- ❖ This value is set in the daemon configuration settings via the `maxNCLientsPerServer` parameter.

**Reusable** – This is an extension of the single client mode. Once the client processing finishes, the server process does not die and can be used by another client, reducing startup times and application startup overhead.

This mode does not have the high overhead of single client mode since the servers are already initialized. However, this server mode may use a lot of server resources (since it requires as many server processes as concurrent clients).

- ❖ When accessing Informix through XA, do not specify this mode. In this case, define a new workspace for Informix, so that all the other data sources you are accessing use reusable servers.

The other modes can be set so that the server processes are reusable. The number of times a process can be reused is controlled by the “Reuse Limit” field value in the Configuration Manager (the maximum number of times a particular server process can be reused or how many clients it can serve before it is retired). Reuse of servers enhances performance since it eliminates the need to repeat initializations. However, reuse runs a risk higher memory utilization over time. The default for the “Reuse Limit” field value is 0, indicating that no reuse limit is enforced.

Set the server mode in the WS Server of the Daemon Editor:

Workspace server mode: multiClient

Reuse limit:  
☐ None  
☒ Maximum: 20

Clients per server limit:  
☒ None  
☐ Maximum: 0

Server availability:  
Initial number of servers: 5  
Minimum number of available servers: 5  
☐ Keep when daemon ends  
☐ Set maximum number of servers: 0

Resource limitations:  
☐ Limit number of active servers: 0

Server priority:  
☒ Use default priority  
☐ Use priority specified: 0

Daemon Control | Daemon Logging | Daemon Security | WS Info | **WS Server** | WS Logging | WS Security | Source

When using any of the server modes you can specify prestarted servers. Prestarted servers are server processes started when the daemon starts. These prestarted servers are then available for use by new client processes, saving initialization time. Instead of starting a new server process each time one is requested by a client, the client receives a

process immediately from a pool of available processes. When the client finishes processing, this server process either dies, or if reusable servers have been specified, it is returned to the pool of available servers.

Prestarted servers are set in the Server Availability section of the WS Server tab.

You set prestarted servers by specifying the following parameters:

**Initial number of servers** – The number of server processes that are prestarted for this workspace when the daemon starts up. These are available for use by new client processes with minimal initialization time. Instead of starting a new server process each time one is requested by a client, the daemon immediately allocates (to the client) a server from a pool of available servers. When the number of available server processes drops below the value specified in “Minimum number of available servers” field, the daemon again starts server processes until the specified number of available servers is reached. The default for this parameter is 0, meaning that no servers are prestarted for this workspace.

**Minimal number of available servers** – The minimum number of server processes in the prestarted server’s pool before the Attunity Connect daemon resumes creating new server processes (up to the number specified in the “Initial number of servers” field value, described above). If this parameter is set to a value greater than that of the “Initial number of servers” field value, the daemon considers the value to be the same as the value specified in the “Initial number of servers” field value (thus, a new server process is started each time one is allocated to a client). The default for this parameter is 0, meaning that new servers are created only when there are no other available servers.

**Set maximum number of servers** – The maximum number of available server processes pooled for this workspace. If the server is reusable, once a client disconnects from the server, the daemon returns the server

to the pool of available servers. If the limit is reached, excess server processes are discarded.

- ❖ This value is set in the daemon configuration settings via the `maxNAvailableServers` parameter.

### OS/390 and z/OS Platforms

In addition to setting up prestarted servers, as described above, you can set prestarted servers for OS/390 and z/OS platforms by specifying this parameter. Thus, setting 10 prestarted servers and 10 prestarted threads results in 100 tasks started (10 threads for each process).

You set prestarted threads in the Resource limitations section of the WS Server tab of the daemon editor

**Number of sub-tasks** – The number of threads for a server that are prestarted for this workspace when the daemon starts up.

### OpenVMS and UNIX Platforms

**Limit number of active servers** – The maximum number of active server processes (either available or in use). Once reached, no new server processes will be created for the particular workspace and client connections would be rejected if there is no available server to accept them. Once the number of active servers drops below the maximum (for example, a client disconnects from a server and the server terminates), new servers can again be started. If this value is set to a non-zero value lower than the “Initial number of prestarted servers” field value, the daemon assumes it is set to the same value as the “Initial number of prestarted servers” field value. The default for this parameter is 0, meaning that no maximum is enforced.

- ❖ Under OpenVMS, the account running the server processes must have SYSLOCK privilege.

### Specifying Which Binding Configuration to Use

If you want the workspace to set a binding configuration other than the default (NAV) configuration, you must specify the new configuration in the relevant Workspace in the daemon configuration.

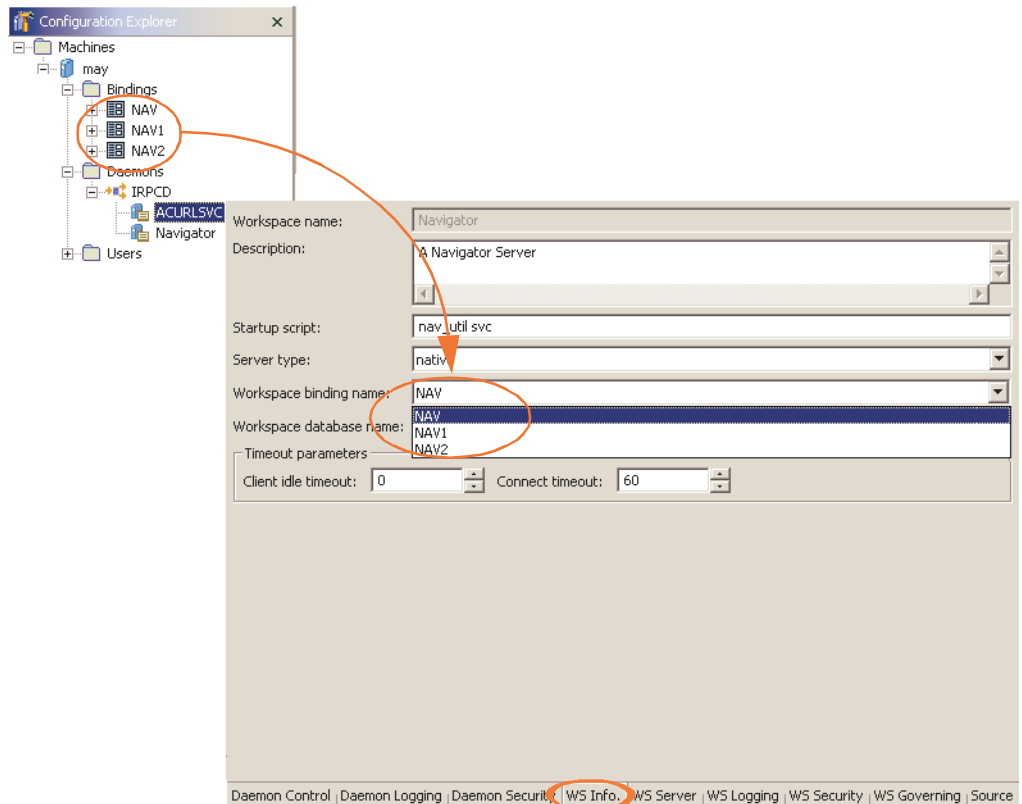
- ❖ You can have different workspaces using different binding configurations.

### OpenVMS and UNIX Platforms

If you want to use a binding other than the default binding, on UNIX and OpenVMS platforms you can specify the binding as part of the startup script for the workspace.

► To specify a binding configuration:

1. Open Attunity Studio by selecting **Start | Programs | Attunity | Studio1.2**.
2. In the Runtime perspective, open the machine where the workspace resides, in the explorer tree.
3. Open the daemon that manages the workspace.
4. Right-click the workspace and select **Edit Workspace Configuration**.
5. In the Workspace editor WS Info tab, select the binding from the list of bindings set on the machine.



## Configuring Daemon Logging

### OS/390 and z/OS Platforms

This section does not apply to OS/390 and z/OS platforms, where logging information is written directly to the process.

The daemon log records the daemon operations, such as RPC calls and error messages. You can select the information you want included in the log file.

### OS/400 Platforms

The AS/400 includes a UNIX file system, which enables file manipulation via standard UNIX commands.

❖ Only a subset of UNIX commands are supported.

Attunity Connect log files are stored under the UNIX file system section of the AS/400.

The following types of files are stored under the native AS/400 file system:

- Executables
- Service programs
- ❖ When manipulating configuration files or log files, you surround the path/filename in single quotes ('), to ensure that the slash (/) used in the UNIX file system syntax is handled correctly. (The slash is an OS/400 special character.)

#### ► To configure a daemon's log file:

1. Open Attunity Studio by selecting **Start | Programs | Attunity | Studio1.2**.
2. In the Runtime perspective, open the machine where the daemon resides, in the explorer tree.
3. Open the daemon you want to log.
4. Right-click the daemon and select the Edit Daemon Configuration option.

5. In the Daemon editor, open the Daemon Logging tab.

6. Enter a name for the Attunity Connect server log file.

- ❖ Under OS/390 and z/OS, the default is to write the log entries to the job only.

The following tokens may appear in the log file template and will be replaced accordingly:

- **%A** – workspace name
- **%D** – date (yymmdd)
- **%I** – instance number of the given workspace server
- **%L** – server account login directory. Under OS/390 and z/OS: The path to NAVROOT.TMP. If you specify a file name without this path, the file is created under 'NAVROOT.TMP.filename'
- **%P** – server process ID
- **%T** – time (hhmmss)
- **%U** – server account name (username)

7. Specify a location for the daemon log data. This parameter must be specified in full and without environment variables.
8. In the Logging Options section, select the Daemon activity to be included in the log.
9. In the Trace Options section, select the parameters whose errors you want to trace.

## Configuring Server Logging

### OS/390 and z/OS Platforms

This section does not apply to OS/390 and z/OS platforms, where logging information is written directly to the process.

The server log records the interaction between the client and the server process, such as RPC calls and error messages. Attunity Connect enables specifying the information you want to include in the log file.

### OS/400 Platforms

The AS/400 includes a UNIX file system, which enables file manipulation via standard UNIX commands.

❖ Only a subset of UNIX commands are supported.

Attunity Connect log files are stored under the UNIX file system section of the AS/400.

The following types of files are stored under the native AS/400 file system:

- Executables
- Service programs
- ❖ When manipulating configuration files or log files, you surround the path/filename in single quotes ('), to ensure that the slash (/) used in the UNIX file system syntax is handled correctly. (The slash is an OS/400 special character.)

#### ► To configure a server's log file:

1. Open Attunity Studio by selecting **Start | Programs | Attunity | Studio1.2**.
2. In the Runtime perspective, open the machine where the workspace resides, in the explorer tree.
3. Open the daemon that manages the workspace.
4. Right-click the workspace and select Edit Workspace Configuration.



5. In the Daemon editor, open the WS Logging tab.

Trace options

☐ No timeout ☐ Extended RPC trace

☐ Call trace ☐ System trace

☐ RPC trace ☐ Timing

☐ Sockets

☒ Specific log file format: Nav\_%.log

Daemon Control | Daemon Logging | Daemon Security | WS Info | WS Server | **WS Logging** | WS Security | WS Governing | Source

6. In the Trace Options section, select the parameters you want to trace.
7. Specify whether you want the log saved in a specific format and specify the format.

You can use the following tokens as part of the name. These tokens are replaced dynamically when the log file is generated:

**%A** – workspace name

**%D** – date (yymmdd)

**%I** – instance number of the given workspace server

**%L** – server account's login directory

**%P** – server process ID

**%T** – time (hhmmss)

**%U** – server account name (username)

For example, a log filename template "%L/server\_%.log" may produce a log file such as: /usr/smith/server\_sales15.log. The default log file template is "%L/**server\_%.log**". (The default can vary according to the operating system. Under OpenVMS, for example, the default is %L**server\_%.log**.)



## Chapter 5 Metadata for a Data Source

Metadata is data that describes the target data. Metadata defines the structure of the data and where it is located. Attunity Connect relies on the native metadata of the data source when connecting to relational data sources (such as Informix, Oracle and Sybase) and some non-relational data sources (such as Adabas, using PREDICT).

Attunity Connect requires its own metadata for data sources whose metadata is not readable by Attunity Connect or which do not have metadata. The data sources that require Attunity Connect metadata (ADD) are:

- Btrieve
- CISAM
- DBMS
- DISAM
- Enscribe
- IMS/DB
- RMS
- VSAM (including VSAM under CICS)
- A virtual database table

Metadata for these data sources can be created from scratch or imported, from sources such as COBOL copybooks or from existing metadata not readable by Attunity Connect (such as a CDD dictionary for RMS metadata). Once the metadata exists in Attunity Connect, it can be edited.

Attunity Connect metadata is imported to Attunity Connect using the Metadata Import perspective and managed using the Metadata perspective.

The native metadata of other data sources, which is used by Attunity Connect (such as relational data sources like Oracle) can also be viewed in the Metadata perspective of Attunity Studio. However, this metadata cannot be edited. Some native metadata does not include information that Attunity Connect requires to fully optimize query execution (for example, the number of rows and blocks in an Rdb table). In this sort of situation you can extend the native metadata by adding these extensions in Attunity Connect metadata.

► **To specify extended metadata for a data source:**

1. Display the metadata for the data source in the Metadata perspective of Attunity Studio.
2. Change the relevant values for the table to be extended in the Statistics tab. The table symbol in the explorer tree is marked with an asterisk to show that the metadata has been extended.
  - ❖ All the information in the other tabs are for reference only and can only be viewed.

Also, you can sometimes improve performance using Attunity Connect metadata instead of the native metadata. In this case, you can export a snapshot of the native metadata to Attunity Connect and use this local copy of the native metadata when accessing the data source.

► **To make a copy of data source metadata:**

1. Display the metadata for the data source in the Metadata perspective of Attunity Studio.
2. Right-click the data source and choose Manage Cached Metadata from the popup menu.
3. Select the tables that you want to use a local copy and move them to the right panel.
4. Click **Finish**. The tables are displayed under the data source.
  - ❖ The table symbol changes from the relational data source symbol to a data source symbol that requires Attunity Connect metadata.

The localCopy property in the data source definition in the binding configuration is set to true.

Only the tables that have cached metadata are displayed in the explorer tree. To revert back to using non-cached metadata, you can either right-click individual tables and choose Delete Cached Table from the popup menu or, for all the tables, right-click the data source and choose Set Metadata followed by Native Metadata from the popup menu.

- ❖ If the native metadata changes from time to time, it is recommended not to use a snapshot of the native metadata.

## Importing Metadata to Attunity Connect

Metadata for Enscribe, IMS/DB, RMS and VSAM can be imported via the Metadata Import perspective of Attunity Studio. Metadata for other data sources can be generated from COBOL copybooks, also within Attunity Studio.

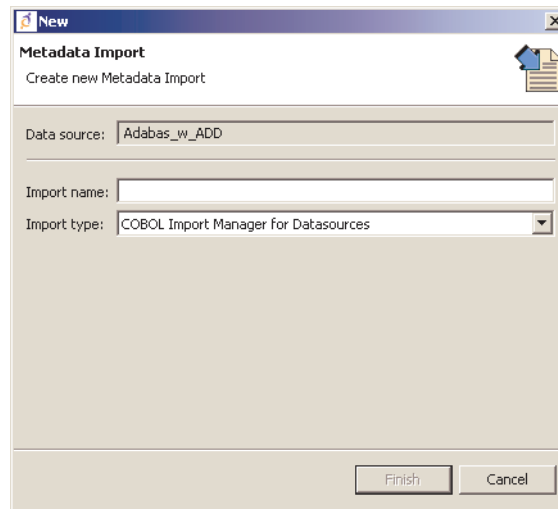
A number of stand-alone import utilities also exist to generate Attunity Connect metadata. For details, refer to *Attunity Connect Reference*.

- ❖ The Enscribe and RMS imports in Attunity Studio are based on COBOL copybooks. If the Enscribe metadata exists in TAL files or on a DDL subvolume, use the stand-alone utilities to generate the Enscribe metadata. If the RMS metadata exists in a CDD data dictionary, use the stand-alone utility to generate the RMS metadata.

## Importing Metadata with Attunity Studio

### ► To import metadata:

1. In the Configuration perspective, right-click the data source you want to import metadata for and select Open import perspective.  
The Metadata Import perspective opens with the selected data source set in the explorer tree.
2. Right-click the data source in the Metadata Import explorer tree and select New Import.
3. In the New Metadata Import window specify the import name and import type and click **Finish**.



A metadata import wizard opens.

The import is dependent on the data source. Thus, some imports have fewer steps than other imports and require different information.

See "Importing Metadata for a Data Source" on page 131 for an example of importing metadata for VSAM.

## Managing Attunity Connect Metadata

► **To set up Attunity Connect metadata:**

1. In the Configuration perspective, right-click the data source whose metadata you want to import.
2. Select Edit Metadata from the popup menu.

The Metadata perspective opens with the selected data source and the tables in it, displayed in the explorer tree.

3. Double-click the table whose metadata you want to manage.
4. In the General tab, table details, such as the name of the table, its physical file location and the way the table is organized is displayed:

The screenshot shows the 'General' tab of the metadata configuration window. At the top, the 'Table name' is 'DOCTOR' and the 'Comment' field is empty. Below this is a large text area for additional comments. The 'Organization' section has two radio buttons: 'Index' (selected) and 'Sequential'. The 'Record length' section has a 'Maximum record length' field set to '0' and a 'DB Command' button. The 'Filter expression' section has a checkbox for 'Set filter expression' which is unchecked, and a text area for the filter expression. At the bottom, there is a tabbed interface with 'General', 'Columns', 'Indexes', 'Statistics', and 'Source'. The 'General' tab is currently selected and highlighted with a red circle.

❖ The fields displayed are dependent on the type of data source.



6. In the Indexes tab, enter index details, such as the name of the index and how the index is ordered:

[illegible]



- 7. In the Statistics tab, enter statistical details, such as the number of rows in the table. The statistical details can be generated by running the update statistics utility (using the **Update** button):

Table

No. of rows:

No. of blocks:

Clear

Update

Columns

Column name	Cardinality
HOSPNAME	0
WARDNO	0
BEDIDENT	0
DOCTNAME	0
DOCT_ADDRESS	0
DOCT_PHONE	0
SPECIALT	0

Indexes

Indexes and segments	Cardinality
<input checked="" type="checkbox"/> HOSPNAME	0
HOSPNAME	0
HOSPNAME - WARDNO	0
HOSPNAME - WARDNO - BEDIDENT	0

General

Columns

Indexes

Statistics

Source

► To set the adapter definition in Attunity Studio:

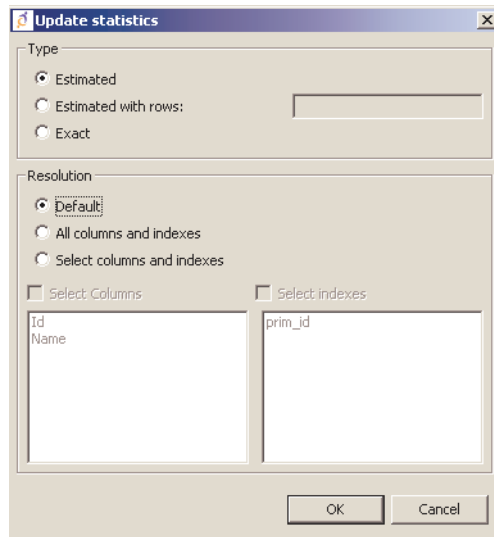
Adapter definitions can be modified to the exact requirements of an application. Interaction definitions are modified Interaction or Interaction General and Interaction Advanced (used for data source adapters) tabs of the interaction editor in the Metadata perspective.

Updating Statistics

Attunity Connect includes a query optimizer, Attunity Connect collects information about tables, indexes, and optionally column cardinalities. These statistics can be used to optimize a query using the Attunity Connect Query Optimizer, which finds the most efficient way to perform a query across multiple machines. This is done relying on metadata statistics.

You can update ADD metadata statistics for a table using the Attunity Studio Metadata perspective. This is done by clicking the **Update** button in the Statistics tab for the table. Clicking **Update** overwrites previous statistics.

Specify the statistics that you want in the displayed screen:

The image shows a dialog box titled "Update statistics". It has two main sections: "Type" and "Resolution". In the "Type" section, there are three radio buttons: "Estimated" (which is selected), "Estimated with rows:" (which has an empty text box next to it), and "Exact". In the "Resolution" section, there are three radio buttons: "Default" (selected), "All columns and indexes", and "Select columns and indexes". Below these are two checkboxes: "Select Columns" and "Select indexes". Under "Select Columns" is a list box containing "Id" and "Name". Under "Select indexes" is a list box containing "prim\_id". At the bottom of the dialog are "OK" and "Cancel" buttons.

where:

**Type** – The type of statistic information being added.

**Estimated** – An estimation of the amount of statistical information returned.

**Estimated with Rows** – An estimation of the amount of statistical information returned. Including an estimation of the number of rows in the table. Specify the number in the text box. This number is used to shorten the time to produce the statistics, assuming that the value specified here is the correct value, or close to the correct value.

- ❖ When the number of rows in the table is not provided, the number of rows used is determined as the maximum value between the value specified in the <tuning DsmMaxBufferSize> environment property and the value set in the nRows attribute (specified as part of the metadata for the data source).

**Resolution** – The level of the statistical information returned.

**Exact** – The exact statistical information returned. Note that this can be a lengthy task and can lead to disk space problems with large tables.

**Default** – Only information about the table and indexes is collected. Information for partial indexes and columns is not collected.

**All Columns and Indexes** – Information about the table, indexes, partial indexes and columns is collected.

**Select Columns and Indexes** – Enables you to select the columns and indexes you want to collect statistics for. In the enabled list of columns and/or indexes left click those columns you want included (you can use shift-click and control-click to select a number of columns and/or indexes).

The statistics are updated on the server, enabling work to continue in Attunity Studio. A message is displayed in Attunity Studio when the statistics on the server have been updated.

- ❖ You can update statistics for more than one table at a time by using the UPDATE\_STATISTICS utility. For details, see *Attunity Connect Reference*.

## Creating Procedure Metadata

You create and edit procedure metadata using Attunity Studio.

### ► To define the procedure in Attunity Studio:

1. Right-click the procedure in the Configuration perspective explorer tree and choose Edit Metadata from the popup menu.

The Metadata perspective opens displaying the procedure.

2. Right-click Procedures node under the procedure and choose New Procedure. The New procedures window opens.
3. Enter the procedure name that the driver will search for in the given DLL.
4. Enter the path and/or name for the DLL or click **Browse** to find the DLL.

The information specified is handled differently by the different supported platforms, as follows:

- On Windows platforms this value specifies the physical filename with neither the dll extension nor the directory portion of the pathname (it should be in a directory in the PATH environment variable, or in %NAVROOT%\bin).
- On OpenVMS platforms, if you define a logical, include the full path and filename. If the system finds the logical, the logical is

- used. If the logical is not found, the system looks for sys\$share:<filename>.exe, or in NAVROOT:[bin].
- On UNIX platforms, this attribute can specify an environment variable. Attunity Connect converts the environment variable name to uppercase before the system performs the lookup. If the system does not find the environment variable, it looks for \$NAVROOT/lib/<filename>.<extension>, where <filename> is replaced by the unconverted value of the filename attribute and <extension> is the shared library extension for the particular UNIX platform.
  - On HP (Compaq) NonStop and OS/390 or z/OS platforms, this attribute specifies a symbol that is used by nav\_register\_function to statically link the Attunity Connect procedure with the Attunity Connect library.
5. Select the source language of the DLL.
  6. Click **Finish**. The procedure appears in the Metadata explorer tree and the procedure editor opens.

**Defining Return Values** Return values for the procedure or specified in the General tab:

Procedure Name: MATH1

Path: Prc\_Samples.dll Browse...

Language: C

Return Value

Name	Data Type	DB Command
Void	Void	

Add  
Delete

General Arguments Source

► **To specify return values:**

1. Click **Add**. The New Field window opens.
2. Enter the return value name and click **OK**.
3. The default return value data type is string. Edit this value by clicking the Data Type and selecting a data type.

The following default return value properties are displayed for each argument:

**MECHANISM** – The method by which this argument is passed/received by the procedure. Valid values are VALUE, DESCRIPTOR, and REFERENCE.

For outer-level (non-nested) arguments, structure arguments (for the structure itself, and not structure members), and variant arguments, the default value is REFERENCE. The default for all other columns is VALUE. A DESCRIPTOR can only be a static descriptor containing strings.

**AS/400 Platforms**  
Any parameter that is an argument to the function (that is, contains a dbCommand statement with a non-zero ORDER) cannot have a MECHANISM of VALUE. This arises from the fact that the natural size of an integer on the AS/400 stack is smaller than a pointer.

❖ This entry is not case sensitive

**ORDER** – The ORDER for a return value is zero (0).

Additional return value properties can be manually added by clicking the Value for a property, and clicking the ellipses button. For details of these values and how to set them, refer to *Attunity Connect Reference*.

Defining Input and Output Arguments

Input and output arguments for the procedure or specified in the Arguments tab:

Input output Parameters

Name	Type	Data Type

Add

Up

Down

Delete

Properties

Property	Value

General

Arguments

Source

► **To specify arguments:**

1. Click Add. The New Field window opens.
2. Enter the argument name and click **OK**.
3. Click the Type field and set the type of the argument (input, output or input/output).
  - ❖ If a argument is defined as Input/Output, an additional argument is created in order to set the parameter's input properties.
4. The default argument data type is string. Edit this value by clicking the Data Type area and selecting a data type.
5. Use the **Up** and **Down** buttons to set the order of the arguments.

The following default argument properties are displayed for each argument:

**MECHANISM** – The method by which this argument is passed/received by the procedure. Valid values are VALUE, DESCRIPTOR, and REFERENCE.

For outer-level (non-nested) arguments, structure arguments (for the structure itself, and not structure members), and variant arguments, the default value is REFERENCE. The default for all other columns is VALUE. A DESCRIPTOR can only be a static descriptor containing strings.

**AS/400 Platforms**

Any parameter that is an argument to the function (that is, contains a dbCommand statement with a non-zero ORDER) cannot have a MECHANISM of VALUE. This arises from the fact that the natural size of an integer on the AS/400 stack is smaller than a pointer.

- ❖ This entry is not case sensitive

**ORDER** – The procedure's argument number. The order can be changed using the Up and Down buttons.

Additional argument properties can be manually added by clicking the Value for a property, and clicking the ellipses button. For details of these values and how to set them, refer to *Attunity Connect Reference*.

## Chapter 6 Metadata for an Application Adapter

Every application adapter requires an adapter definition. This definition is set up and maintained in Attunity Studio as metadata.

### Importing Adapter Metadata

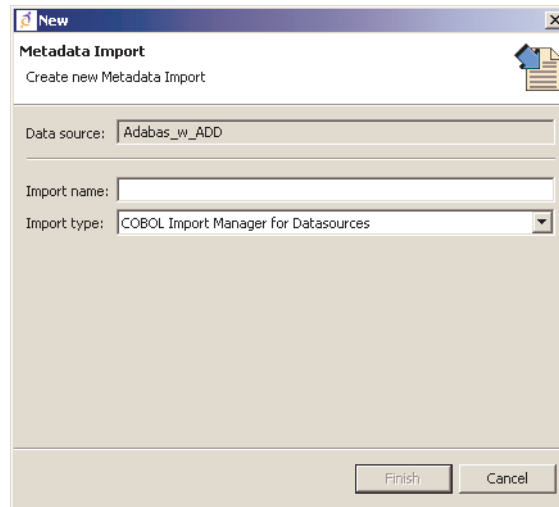
Metadata for CICS, IMS/TM and Tuxedo can be imported using tailored import utilities in Attunity Studio. All other adapters that have COBOL copybooks describing the input and output structures used by the application can use a generic import in Attunity Studio. All imports are performed via the Metadata Import perspective of Attunity Studio.

► **To import metadata:**

1. In the Configuration perspective, right-click the adapter for which you want to import the metadata and select Open import perspective from the popup menu.

The Metadata Import perspective opens with the selected adapter displayed in the explorer tree.

2. Right-click the adapter.
3. In the New Metadata Import window specify the import name and import type and click **Finish**.



A metadata import wizard opens.

The import is dependent on the adapter. Thus, some imports have fewer steps than other imports and require different information.

See "Importing Metadata for an Adapter" on page 143 for an example of importing metadata for a CICS adapter.

## Managing Metadata with Attunity Studio

► To set up adapter metadata in Attunity Studio:

1. In the Configuration perspective, right-click the adapter whose metadata you want to manage.
2. Select Edit Metadata from the popup menu.

The Metadata perspective opens with the selected adapter and its interactions, displayed in the explorer tree.

3. Double-click the adapter whose metadata you want to manage.
4. In the General tab, specify details, such as the way in which you connect to the adapter.

[illegible]





**Description** – An optional description of the interaction.

**Mode** – The interaction mode:

- **sync-send-receive** – The interaction sends a request and expects to receive a response.
- **sync-send** – The interaction sends a request and does not expect to receive a response.
- **sync-receive** – The interaction expects to receive a response.

**Input/Output Definitions** – Identifies the input and output records.

**Interaction Specific Parameters** – Specific properties for the interaction.

6. In the Schema General tab, specify the general attributes of the adapter.

The screenshot shows a software interface for configuring a schema. At the top, there's a 'Details' section with three text input fields labeled 'Name:', 'Version:', and 'Header:'. Below this is a large, empty rectangular area. At the bottom, there's a horizontal navigation bar with several tabs: 'General', 'Interaction General', 'Interaction Advanced', 'Schema General' (which is highlighted with an orange oval), 'Schema Record', and 'Source'.

where:

**Schema Name** – The name of the adapter.

**Version** – The schema version.

**Header** – A C header file with the data structures for the adapter. This header file is used when the C API to applications is used.

7. In the Schema General tab, specify the grouping of fields.

**Fields list** – Defines the single data item within a record.

- a) Use the New Field button to add fields and the Delete button to delete the selected field as needed.

**Name** – The name of the field

**Type** – The data type of the field. The following are valid data types:

Binary	Int
Boolean	Long
Byte	Numeric[(p,s)]
Date	Short
Double	String
Enum	Time
Float	Timestamp

**Length** – The size of the field including a null terminator, when the data type supports null termination (such as the cstring data type).



## Chapter 7    Metadata for an Event

The event adapter creates a queue for incoming events. The adapter definition for an event adapter describes the interactions that enter the queue. This definition is set up and maintained in Attunity Studio, in the Metadata perspective.

If COBOL copybooks describing the event exist, the metadata can be generated from the copybooks, using the Attunity Connect Metadata Import perspective.

### Creating Metadata

The following procedure describes how to create event metadata when a COBOL copybook is not available.

► **To create adapter metadata:**

1. In the Configuration perspective, right-click the event whose metadata you want to create.
2. Select Edit Metadata from the popup menu.

The Metadata perspective opens with the selected event displayed in the Explorer tree.

3. Right-click the Schema node and choose New record from the popup menu.
4. Enter a name for the new record definition.

5. Click **OK**. The editor opens with the new record definition displayed.

The screenshot shows a software interface for defining record specifications. It is divided into two main sections: 'Fields List' and 'Specifications'.

**Fields List:** This section contains a table with three columns: 'Name', 'Type', and 'Length'. The first row is pre-filled with 'Rec1' in the 'Name' column and 'Record' in the 'Type' column. The 'Length' column is empty. To the right of the table are two buttons: 'New Field' and 'Delete'.

**Specifications:** This section contains a table with two columns: 'Property' and 'Value'. The table is currently empty, with multiple rows available for input.

**Navigation Bar:** At the bottom of the window, there is a horizontal navigation bar with five tabs: 'General', 'Interaction', 'Schema General', 'Schema Record', and 'Source'. The 'Schema Record' tab is currently selected.

- Click **New Field** to add new fields for the record, describing the data structure for the interaction.

Fields List

Name	Type	Length
<input type="checkbox"/> Rec1	Record	
field1	string	

New Field  
Delete

---

Specifications

Property	Value
align	
array	
case	
counter	
default	
defaultNotDisplayed	false
filter	
form	false
implicit	false
mechanism	
nativeType	
offset	
paramNum	
precision	
private	false
reference	false
required	false
scale	
selector	
size	

General | Interaction | Schema General | Schema Record | Source

- Continue adding fields until the record structure is complete.

## Managing Metadata with Attunity Studio

An event is a specific case of an adapter and is managed in the same way any other adapter is managed: In the Attunity Studio Metadata perspective.

### ► To edit event metadata:

- In the Configuration perspective, right-click the event whose metadata you want to edit.
- Select Edit Metadata from the popup menu.

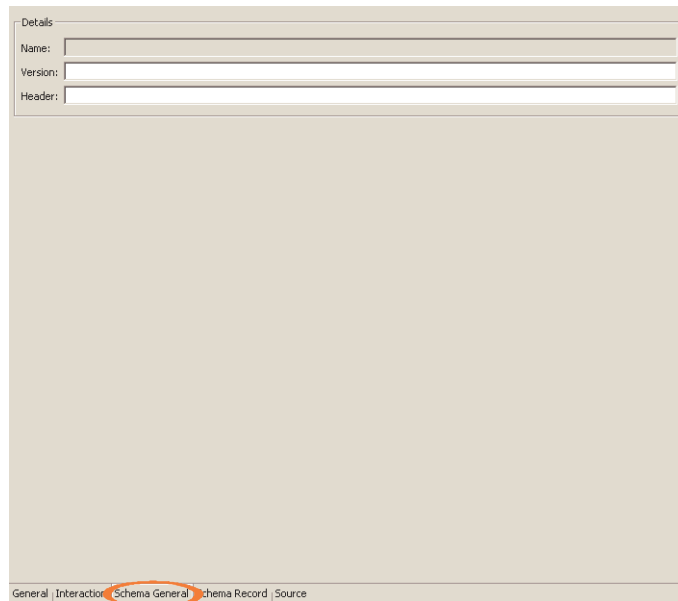
The Metadata perspective opens with the selected event displayed in the Explorer tree.







7. In the Schema General tab, specify the general attributes of the event.



The screenshot shows a software interface with a tabbed view at the bottom. The tabs are labeled 'General', 'Interaction', 'Schema General', 'Schema Record', and 'Source'. The 'Schema General' tab is currently selected and highlighted with an orange oval. The main area of the 'Schema General' tab contains a 'Details' section with three input fields: 'Name:', 'Version:', and 'Header:'. The 'Name' field is the top-most and longest, followed by 'Version' and then 'Header'.

where:

**Schema Name** – The name of the event.

**Version** – The schema version.

**Header** – A C header file with the data structures for the event. This header file is used when the C API to applications is used.

8. In the Schema Record tab, specify the grouping of fields.

The screenshot shows a software interface for defining a schema record. The 'Fields List' section at the top contains a table with three columns: 'Name', 'Type', and 'Length'. The first row is expanded, showing 'field1' of type 'string'. To the right of this table are 'New Field' and 'Delete' buttons. Below this is the 'Specifications' section, which is a large empty table with 'Property' and 'Value' columns. At the bottom, a tab bar shows 'General', 'Interaction', 'Schema Generator', 'Schema Record' (selected and circled), and 'Source'.

**Fields list** – Defines the single data item within a record.

- a) Use the New Field button to add fields and the Delete button to delete the selected field as needed.

**Name** – The name of the field

**Type** – The data type of the field. Following are valid data types:

Binary	Int
Boolean	Long
Byte	Numeric[(p,s)]
Date	Short
Double	String
Enum	Time
Float	Timestamp

**Length** – The size of the field including a null terminator, when the data type supports null termination (such as the cstring data type).



# Chapter 8    Setting Up Security

Security in Attunity Connect can be set at the following levels:

**Design Time Security** – Secures access to resource definitions within Attunity Connect.

- Setting access rights to machines.
- Granting administrative rights to users to manage resource definitions.

**Run Time Security** – Secures connectivity via Attunity Connect.

- Granting users access rights to resources.
- Enabling a connection through a firewall.
- Sending and receiving encrypted data.

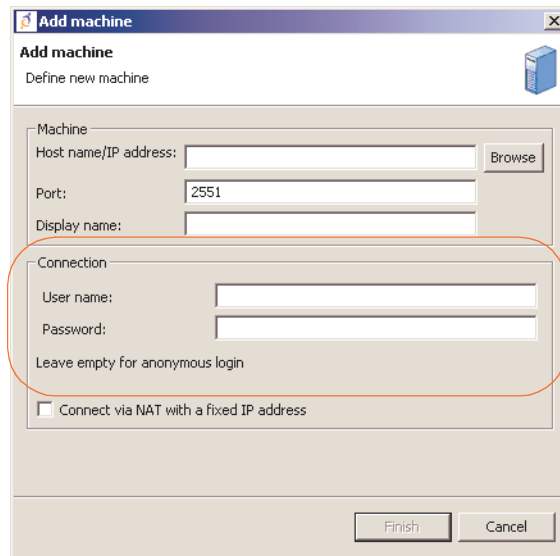
## Configuring Design Time Security

Design time access rights can be restricted to specific users. The following are the restrictions that can be set for design time use:

- Granting initial access to a machine from within Attunity Studio.
- Granting administrative rights to users, including:
  - Authorization to change settings for a specific workspace.
  - Restricting access to the runtime User profile list.

## Initially Accessing the Machine

The first time you connect to a machine via Attunity Studio, the username and password needed to connect to the machine are entered in the Connection section of the Add machine window.



Access to the machine is set by specifying the following information.

**Host name/IP address** – The name of the machine on the network. The name can be entered manually or found using the Browse button, which lists all the machines running an Attunity Connect daemon listener on the specified port currently accessible over the network.

**Port** – Specifies the port where the Attunity Connect daemon is running. The default port for Attunity Connect is 2551.

**Display name** – (Optional) An alias used to identify the machine when different from the host name.

**User name** – (Optional) The username of a user defined as an administrator for the machine.

An administrator is specified when the machine is installed or using NAV\_UTIL ADD\_ADMIN on the machine itself, as described in *Attunity Connect Reference*.

**Password** – (Optional) The password of the user.

## User Authorization Within Attunity Studio

Once a machine is defined in Attunity Studio, you can authorize viewing and editing rights of users and groups according to their role in the design process. Users can be set to the following roles:

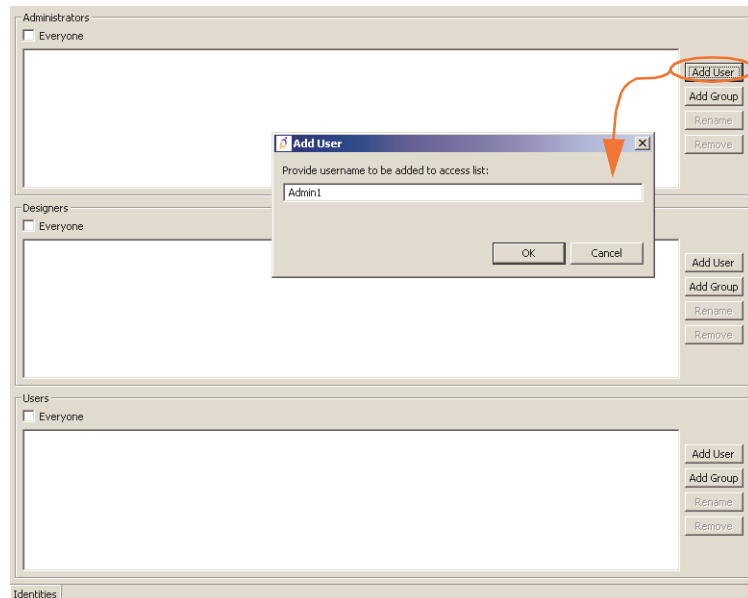
**Administrators** – Allowed to edit all Attunity Studio definitions.

**Designers** – Allowed to edit binding definitions and view daemon definitions in Attunity Studio.

**Users** – Allowed to view the definitions in Attunity Studio.

► **To assign authorization rights to users:**

1. Right-click the machine in the Attunity Studio Configuration perspective, and choose Administration Authorization from the popup menu.
2. In the Administration Authorization window, use the Add User and Add Group buttons to assign authorization to specific users and groups of users.



- ❖ Administration Authorization is set from the top down, meaning that if a higher level is set, there is no need to set the same users for the lower levels.

3. If an authorization level is not restricted to specific users, set it to Everyone.

## Workspace Authorization Within Attunity Studio

Once a machine is defined in Attunity Studio, you can authorize rights to a specific workspace.

► **To assign authorization rights for a workspace:**

1. Right-click the workspace in the Attunity Studio Configuration perspective, and choose Set Authorization from the popup menu.
2. Specify the user name and password for the user for this workspace.

## User Profile Security

Attunity Connect keeps a list of users authorized to accessed machines, data and applications at runtime. This list comprises username and password pairs passed to the target, so that the user does not need prompting. The passwords in this list (a user profile) are encrypted automatically by Attunity Connect. Access to the user profile itself can also be restricted by setting a master password to access the profile.

► **To set a master password for a user profile:**

1. Right-click User in Attunity Studio Configuration explorer and choose Change Master Password from the popup menu.
2. Enter the password to change and the new password and click **OK**.

## Configuring Runtime Security

The following security is available at runtime:

**Daemon and Workspace Administration Rights** – Specifies users who can manage a daemon during runtime.

**Client Access** – Provides ways of ensuring that only authorized users can access the target.

**Passing Through a Firewall** – Specifies ways to connect to a daemon and server process though a firewall.

**Encrypting Network Communications** – Dealing with encrypted information.

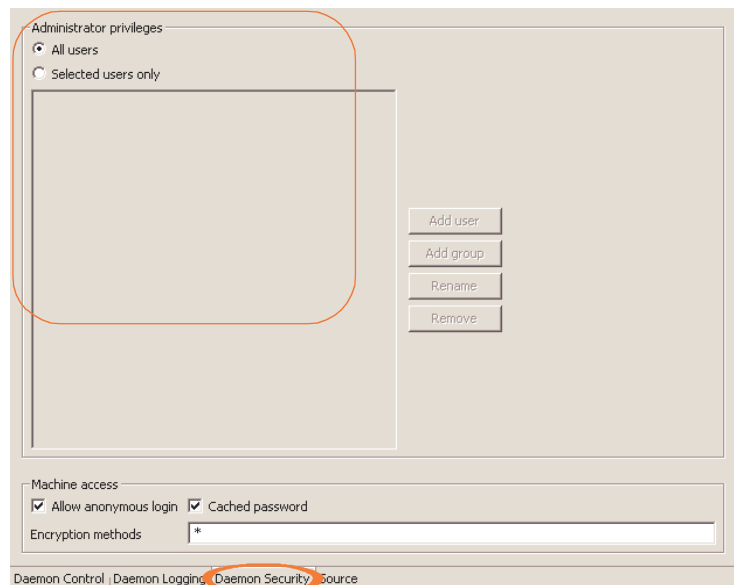


## Daemon and Workspace Administration Rights

Granting administration rights allows a user to manage a daemon and its workspaces. For example, the specified user can refresh a daemon or workspace, or shutdown the daemon.

► **To grant administrative rights to a daemon:**

1. In the Configuration explorer, expand the machine on which the daemon resides.
  - ❖ You can add administrative rights in offline design mode, in a design machine and later drag-and-drop the daemon definition to this machine, as described in "Using an Offline Design Machine to Create Attunity Connect Definitions" on page 37.
2. Expand Daemons and right-click the daemon to set administrative rights for and choose Edit Daemon from the popup menu.
3. Select the Daemon Security tab.



4. Click the Selected users only option.
5. Use the Add Group and Add User buttons to grant Administration rights to groups and users.

### OpenVMS and OS/390 or z/OS Platforms

To define a group of users instead of a single user, preface the name of the group in the configuration with '@'. Under OS/390 and z/OS the group name is validated by a security system such as RACF.

► **To grant administrative rights to a workspace:**

1. Right-click the workspace under the relevant daemon in the Attunity Studio Configuration explorer and choose **Edit Workspace** from the popup menu.
  - ❖ You can add administrative rights in offline design mode, in a design machine and later drag-and-drop the workspace definition to this machine, as described in "Using an Offline Design Machine to Create Attunity Connect Definitions" on page 37.
2. Select the **WS Security** tab.
3. Click the **Selected users only** option.
4. Use the **Add Group** and **Add User** buttons to grant Administration rights to groups and users.

### OpenVMS and OS/390 or z/OS Platforms

To define a group of users instead of a single user, preface the name of the group in the configuration with '@'. Under OS/390 and z/OS the group name is validated by a security system such as RACF.

The screenshot displays the Attunity Studio Configuration Explorer interface. The main window is divided into two sections: 'Workspace access' and 'Administration'. The 'Workspace access' section includes a 'Workspace users' list with 'User1' and 'GroupA' entries, and buttons for 'Add user', 'Add group', 'Rename', and 'Remove'. The 'Selected users only' radio button is selected. To the right, there are checkboxes for 'Enable port range' (unchecked), 'Use specific workspace account' (checked), and 'Allow anonymous client login to server account' (checked). The 'Administration' section, which is highlighted with an orange box, includes an 'Administrator privileges for the workspace' list with 'User2' and 'GroupB' entries, and buttons for 'Add user', 'Add group', 'Rename', and 'Remove'. The 'Selected users only' radio button is also selected here. The 'Add group' button is highlighted with an orange box. The bottom navigation bar shows several tabs: 'Daemon Control', 'Daemon Logging', 'Daemon Security', 'WS Info.', 'WS Server', 'WS Logging', 'WS Security' (highlighted with an orange box), 'WS Governing', and 'Source'.

## Client Access

You can configure the daemon to authorize clients at the following levels:

- Accessing the server machine or target data source or application.
- Accessing the workspace.

### The User Profile

During runtime, access to machines and the target data sources and applications often require valid user names and passwords to be passed as part of the connection information. Attunity Connect enables you to provide this information in a user profile.

A user profile contains the user name and password pair for each machine, data source or application adapter listed in the binding.

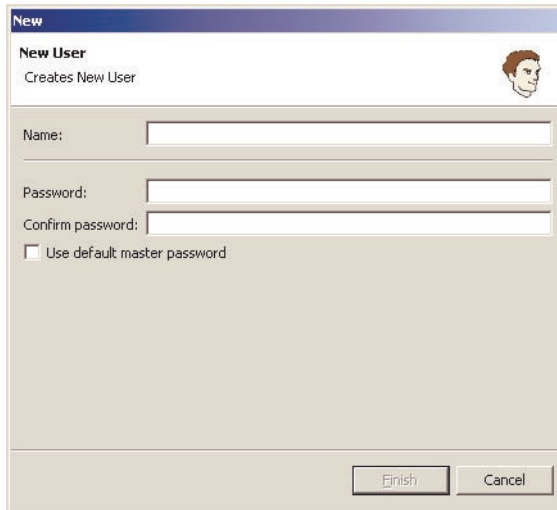
The user profile used when connecting is the profile with the same name as the binding configuration used. Thus, if the default NAV binding is used, the NAV user profile is also used.

User profiles are managed in the Configuration perspective of Attunity Studio.

#### ► To add a new user profile:

1. Open Attunity Studio (accessed by selecting **Start | Programs | Attunity | Studio1.2**).
2. In the Configuration explorer, open the machine the you want to add the user profile for.
  - ❖ You can add a user profile in offline design mode, in a design machine and later drag-and-drop the user profile to this machine, as described in "Using an Offline Design Machine to Create Attunity Connect Definitions" on page 37.
3. Right-click users and select **New User** from the popup menu.
4. In the New Users window enter a name for the user.

5. Enter the password used to secure the user profile.

A screenshot of a 'New User' dialog box. The title bar is blue with the word 'New' in white. Below the title bar, the text 'New User' is displayed in bold, followed by 'Creates New User' in a smaller font. To the right of this text is a small cartoon icon of a man's head. The main area of the dialog box is light beige and contains three text input fields: 'Name:', 'Password:', and 'Confirm password:'. Below these fields is a checkbox labeled 'Use default master password'. At the bottom right of the dialog box are two buttons: 'Finish' and 'Cancel'.

6. Click **Finish**.

The New User editor is displayed enabling authenticators to be set for each of the resources (data sources, application adapters and machines) accessed using the profile.

► **To add users for a resource in the user profile:**

1. In the Configuration explorer, right-click the user under the Users directory and choose Edit User from the popup menu.

The Users profile list opens.



3. In the Workspace Access section, click the Selected users only option.
4. Add Workspace Users (accounts) for users who can access data using this workspace:

## OpenVMS and OS/390 or z/OS Platforms

To define a group of users instead of a single user (so that the group name is validated by a security system), preface the name of the group in the configuration with '@'.

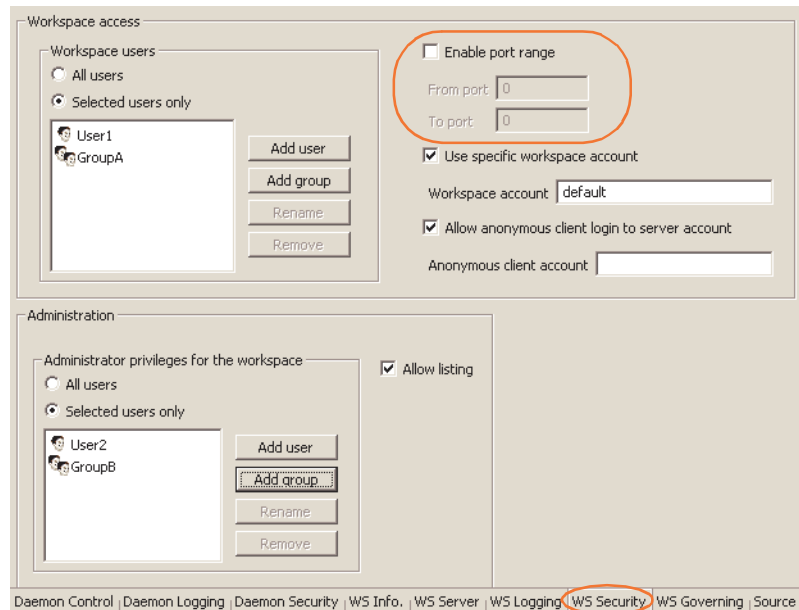
- ❖ When you restrict access to a specific workspace (by unchecking the Anonymous Client Allowed check box and specifying the users via the Workspace Access field), you must also define a user profile on the machine with the workspace for each user defined as able to access the workspace. Each user profile must have the same name as specified in the Workspace Users field. For details of user profiles, see "The User Profile" on page 99.

## Passing Through a Firewall

Attunity Connect enables you to access a server through a firewall in the following ways:

- Specifying a range of ports through which Attunity Connect can access the server.
- Using a SOCKS server.

- When the server uses NAT.
- **To specify a port range:**
  1. Right-click the workspace in the Attunity Connect Configuration perspective and choose Edit Workspace from the popup menu.
  2. Select the WS Security tab.
  3. Specify the port range that is allowed for use with Attunity Connect servers. Only access to these ports is then available. Make sure that the system enables access through the firewall for these ports:




## Encrypting Network Communications

To encrypt communications passed over the network, you need to specify the following encryption parameters in Attunity Connect:

- On the client machine:
  - Set the encryption protocol.
  - Specify that information is encrypted in the user profile.
- On the server machine:
  - Set the encryption key.

► To set the encryption protocol for the server machine:

- [illegible]

- 
- Network settings
- Encryption Protocol: NONE
- Firewall Protocol: NONE
- OK Cancel
- PRIVATE  
DES3  
NONE  
RC4  
RLE



### Specifying Encrypted Communication

After specifying the encryption protocol, you must specify which servers the client is going to communicate with using the specified encryption protocol.

- **To specify the server machine that the client communicates with via encryption:**
  1. Right-click the user profile in the Attunity Connect Configuration perspective and choose Edit User from the popup menu.
  2. In the User editor, select the client machine and click **Add** to open the Add Authenticator window.
  3. Specify the following in the Add Authenticator window:

The screenshot shows a Windows-style dialog box titled "Add Authenticator". It contains two main sections. The first section, "Resource information", has a "Resource type" dropdown menu set to "Remote machine" and a "Resource name" text field containing "enckey:hp.acme.com". The second section, "Authentication information", has three text fields: "User name" with "scott", "Password" with "\*\*\*\*\*", and "Confirm password" with "\*\*\*\*\*". At the bottom right are "OK" and "Cancel" buttons.

**Resource type** – Specify the resource type as Remote machine.

**Resource name** – Specify the communication to the machine is encrypted in the following format:

`enckey:machine_name`

where *machine\_name* is the machine you are connecting to.

**username** – The name optionally associated with the encryption password and which the daemon on the remote machine looks up.

Multiple clients may specify this name in their user profile. In this case, the user profile on the remote machine needs to list only this one username/password entry for network encryption (rather than listing and looking up multiple username/password pairs).

If this username entry is not specified, the daemon on the remote machine uses the name of the current active user profile.

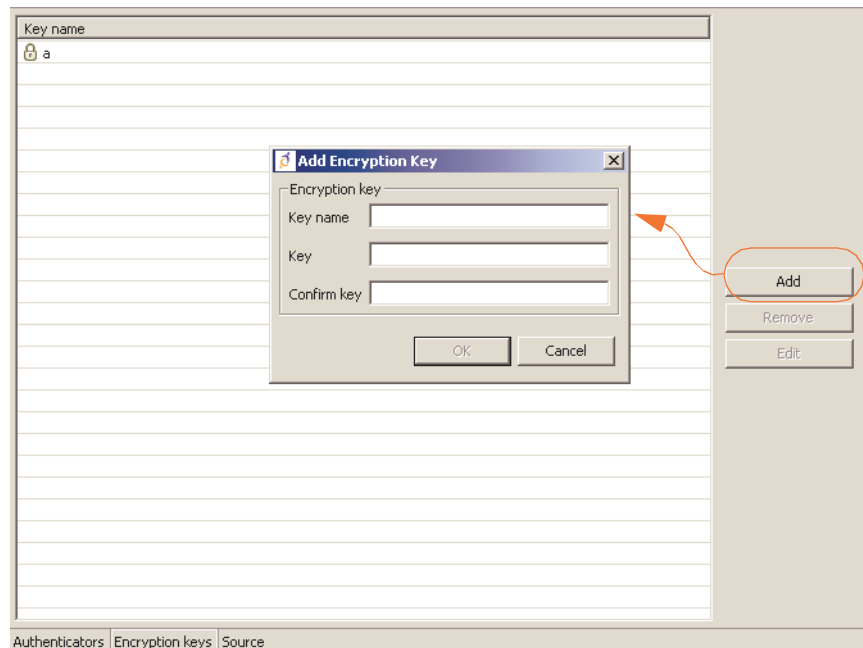
**password** – The password that is required in order to pass or access encrypted information over the network.

**Specifying the Encryption Key on the Server Machine**

Any communication from the client to the server machine is encrypted. The server machine must be configured to decipher the encrypted information using an encryption key.

► **To specify the encryption key on the server machine:**

1. Select the server machine in the Attunity Connect Configuration perspective.
2. Right-click the user profile in the Attunity Connect Configuration perspective and choose Edit User from the popup menu. The User editor opens.
3. Open the Encryption Key tab.



4. Click Add and specify the following in the Add Encryption Key window:

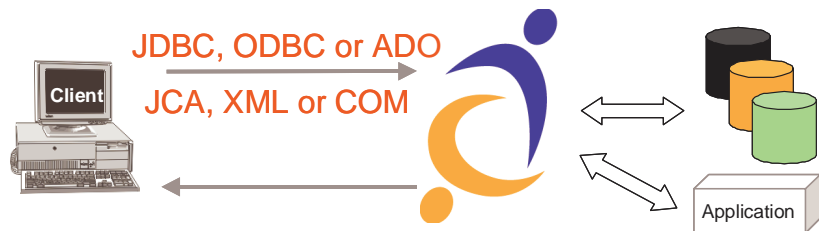
**Key name** – Enter the name associated with the encryption password and which the daemon on this machine looks up.

**Key** – Enter the Encryption key.

**Confirm key** – Re-enter the Encryption key.

## Chapter 9 The Connect String From an Application to Attunity Connect

In order to access data or an application using Attunity Connect, you specify a connect string to Attunity Connect. The binding information of the data source or application specified in the binding configuration is then used to access the target.



To access Attunity Connect, you specify Attunity Connect as the provider in the connect string in the application.

The content of the connect string depends on the API you are connecting through: JDBC, ADO (OLE DB), ADO.NET, or ODBC for data and JCA, .NET, XML or COM for applications.

For a full description of the available parameters for connect strings, refer to *Attunity Connect Reference*.

### The JDBC Connect String

Before using Attunity Connect to connect via the JDBC connection string, you need to set up the appropriate JDBC driver and set its classpath environment variable. For details, see Attunity Connect Thin Client documentation.

Load and register the Attunity Connect JDBC driver by invoking the method `Class.forName` in Java, supplying the classname as follows:

```
Class.forName ("com.attunity.jdbc.NvDriver");
```

- ❖ The setting in the CLASSPATH variable specifies the version of JDBC that you want to run.

To connect to Attunity Connect you call the `getConnection` method and supply a JDBC connection string.

The basic connection string for the `getConnection` method has the following syntax:

```
jdbc:attconnect://machine[:port]
```

**machine** – The IP address where the Attunity Connect daemon runs.

**port** – The port that the daemon listens to. If omitted, the default (2551) is used.

### Example

```
Jdbc:attconnect://dev.acme.com;
```

## The ODBC Connect String

Connecting to Attunity Connect through ODBC requires that you set up a DSN. On Windows platforms, this is created using the ODBC Data Source Administrator.

On non-Windows platforms, the DSN must be a name of a data source defined in the Attunity Connect binding information.

After a DSN is defined, there are two ways to connect to Attunity Connect through ODBC:

- Passing to the application the name of the DSN and (if required) a username and password.
- Using the `SQLDriverConnect` method and passing a connect string.

To work with Attunity Connect as the ODBC provider, supply one of the following in the connect string:

```
DRIVER=Attunity Connect Driver
```

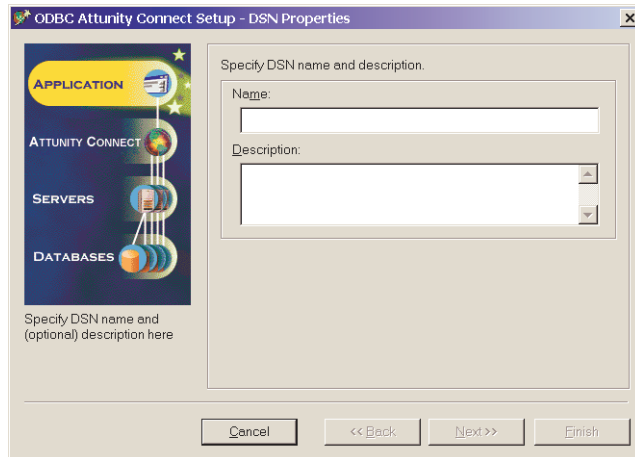
or

```
DSN=name_of_DSN
```

where *name\_of\_DSN* is the name of a data source defined in the binding information.

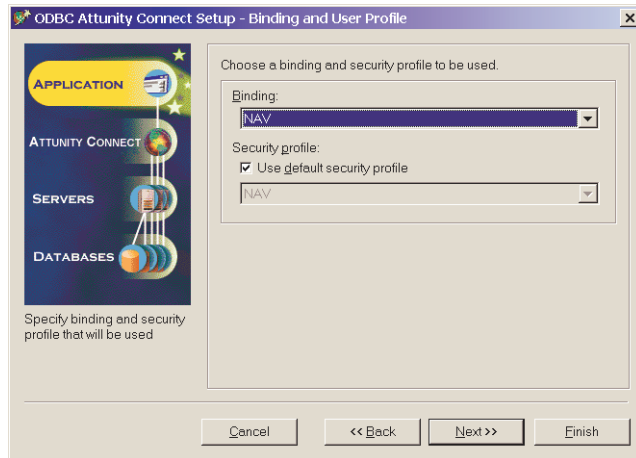
► **To define a DSN on Windows platforms:**

1. Access the Microsoft ODBC Data Source Administrator from the Control Panel (for example, via Windows 2000 it is accessed via Start | Settings | Control Panel | Administrative Tools | Data Sources (ODBC)).
2. Choose the type of DSN you want to define (a User, System or File DSN).
3. Click **Add...**. The Create New Data Source window is displayed.
4. Select Attunity Connect Driver from the list and click **Next** or **Finish** (depending on whether you are defining a File DSN or User or System DSN).
5. The following window is displayed:

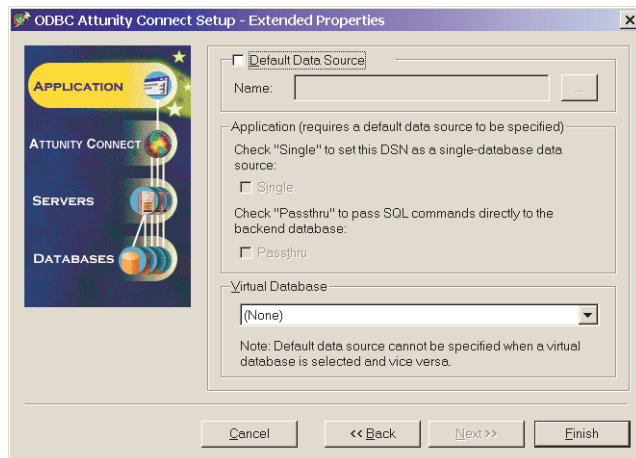


6. Specify a name for the DSN and, optionally, a description.

7. Click **Next**. The following window is displayed:



8. Specify the binding configuration and user profile to be used with the DSN.
9. Click **Next**. The following window is displayed:



10. Specify the properties for the DSN.
11. Click **Finish**.
- ❖ Other properties can be specified by editing the DSN as a text file.

## The ADO Connect String

To specify Attunity Connect as the provider through ADO, use the Open method of the Connection object. You can connect to Attunity Connect as your provider using a Microsoft UDL, as follows:

```
connection.Open "file name=UDL_filename"
```

where *UDL\_filename* is the full path of the UDL file, a Microsoft universal data link file.

You can alternatively use the following format to connect to Attunity Connect:

```
connection.Open "provider=AttunityConnect"
```

## The XML Connection Document

Using an XML document with the Attunity Connect XML protocol (ACX) you can direct requests and receive responses to and from Attunity Connect.

The following ACX shows a basic XML document, with the minimum requirements needed by Attunity Connect.

```
<?xml version="1.0"?>
<acx>
  <connect adapter="adapter_name" />
    <execute>
      <interaction>
        ...
      </interaction>
    </execute>
  <disconnect/>
</acx>
```

The above ACX is constructed as follows:

**The connect Verb** – The connect verb establishes a connection context to the specified adapter. You can, optionally, set the adapter to run on a particular workspace by defining the name of the adapter and the workspace where the adapter runs:

```
<connect adapter="workspace/adapter_name"
... />
```

where:

**workspace** – The name of the workspace where the adapter runs.

**adapter\_name** – The name of the adapter.

**The execute Verb** – The execute verb executes the specified interaction against the adapter and, if appropriate, produces an output record as the return value.

**Interaction** – An interaction to the application adapter, defined in the adapter definition.

**The disconnect Verb** – Closes the current connection and releases all the resources associated with the current connection.

### Example

The following is an example XML document using the query adapter and query interactions to execute a select statement.

```
<?xml version="1.0"?>
<acx>
  <connect adapter="query" />
  <execute>
    <query id="1">
      select * from navdemo:nation
    </query>
  </execute>
  <disconnect/>
</acx>
```

### The Interaction Response

The response returned is an XML document of the following format:

```
<?xml version="1.0"?>
<acx type='response">
  ...
</acx>
```

## The JCA Connection

Attunity Connect supports the following interfaces:

Interface Summary	
AttuDom	Interface for all XML based objects.
AttuInteractionMd	Describes the metadata of the interaction object.



Interface Summary	
AttuMdItem	Common interface for all Attunity Connect metadata objects: <ul style="list-style-type: none"> <li>■ AttuInteractionMd</li> <li>■ AttuResourceMd</li> <li>■ AttuEnumMd</li> <li>■ AttuRecordMd</li> <li>■ AttuFieldMd</li> </ul>
AttuResourceMd	Metadata for the application adapter (EIS)

Attunity Connect includes a number of classes that can be used in the JCA implementation to access an application via an Attunity Connect application adapter.

Complete online documentation of the Attunity-specific JCA extensions is available in HTML format and is accessible via [index.html](#), which resides in `Java\ResourceAdapter\doc`, under the directory where Attunity Connect is installed. Further documentation is also available in Attunity Connect Thin Client documentation.

### Example

The following snippet of code shows the connection to Attunity Connect with JCA. Attunity Connect JCA classes and interfaces are bolded throughout the text.

```
...
import com.attunity.adapter.*;

...
public static void main(String[] args) throws
ResourceException, java.io.IOException
{
    AttuManagedConFactory mcf = new AttuManagedConFactory();

    // Ask user for server, user name and password information
    configureEis(mcf);
    String sSql = askUser("Enter the SELECT statement for
    execution:");

    AttuConnectionFactory cf =
    (AttuConnectionFactory)mcf.createConnectionFactory();
    //-----// 20 sec
    cf.setTimeout(20000);
}
```

```
try {  
    Connection con = cf.getConnection();  
    Interaction interaction = con.createInteraction();  
    ...  
}
```

## The COM Connection

Attunity Connect includes a COM object, called ComACX, that enables accessing Attunity Connect from any COM-based application.

For example, you can use ComACX to integrate a variety of legacy systems and data sources with Microsoft BizTalk Server projects transparently.

The Attunity Connect ComACX object accepts an XML document and replies with an XML document.

The ComACX object must be installed on the same machine that runs the COM-based application. However, Attunity Connect itself does not need to be installed on this machine.

If Attunity Connect is installed on the Windows machine, ComACX is automatically registered as part of the Attunity Connect installation.

### ► To install the ComACX object on another machine:

1. Copy the ComACX object from NAVROOT\bin to a directory on the machine that runs the COM-based application.
  - ❖ NAVROOT is the directory where Attunity Connect is installed.
2. Register the ComACX object on the Windows platform by entering the following in **Start | Run**:

```
regsvr32 fullpath\ComACX.dll
```

where *fullpath* is the full path to the file.

- ❖ Attunity Connect **does** have to be installed on the server machine where the legacy system or data source is located.

The ComACX methods automatically establish a connection with Attunity Connect. An explicit connection can be made using the Connect method.

# Chapter 10 Using Attunity Connect SQL Enhancements

Within an application, Attunity Connect accepts the following versions of SQL to access data:

- SQL specific to the data source you want to access.
- Standard ANSI '92 SQL.

Additionally, you can use the Attunity Connect SQL enhancements. These enhancements include joining data from multiple different data sources within a single SQL query.

This chapter describes the following enhancements:

- Executing multiple queries and update statements in a single SQL statement (see page 115).
- Joining data from different data sources in a single query (see page 116).
- Handling hierarchically structured data and arrays (see page 116).
- Copy data from one table to another in a different data source (see page 120).
- Passing queries directly to the data source (see page 121).

## Batching SQL Statements

You can process multiple queries within an ADO or ODBC application by batching the SQL statements, one after the other, separated by semi-colons (;), as in the following example:

```
sql1;sql2;sql3
```

Parameters are passed as a group for all the queries in the same order as they appear in the individual queries.

- ❖ You cannot use this syntax to batch SQL statements from a Java application or via the NAV\_UTIL EXECUTE utility or ADO Demo application supplied with Attunity Connect.

### ADO Notes

- Set the Multiple Results connection property to enable multiple results (before executing the compound query):  
`oConn.Properties("Multiple Results") = 1`
- The results of the first query are displayed. To see the results of the next query, request `NextRecordSet`.

## Joining Data from Multiple Data Sources in a Single Query

In the SQL, differentiate between data sources by using the data source names that are assigned to the data sources in the binding settings. Separate this data source name and the table name with a colon (:).

### Example

The following example joins data from an Oracle relational database and an RMS file system:

```
SELECT au_lname, au_fname from Oracle:authors
  where city = ANY
    (SELECT city FROM rms:publishers)
```

## Hierarchical Queries

A hierarchical query is a query whose result is a hierarchy of rowsets reflecting parent-child relationships.

In Attunity Connect, rowsets with arrays of structures as columns (which are supported by certain providers) are modeled such that the rows of an array constitute the children of a column in the containing parent row.

Hierarchical queries enable you to do the following:

- Arrange rowsets resulting from a query in a hierarchy, reflecting a parent-child relationship. Use nested `SELECT` statements to do this (see page 117).
- Manipulate data that is stored hierarchically in a data source (such as information stored in arrays in RMS). See page 118.

Currently Attunity Connect supports arrays in the Adabas, CISAM, DISAM, DBMS, Enscribe, RMS, and VSAM drivers.

You can handle this type of data in the following ways:

- By including the hierarchical data as a parent-child relationship (see page 117).

- By flattening the hierarchical data (see page 118).
- By using virtual tables to represent the data (see page 119).

## Generating Hierarchical Results Using SQL

A hierarchical query *nests* a SELECT statement as one of the columns of the rowset retrieved by a nested SELECT statement. You use braces ({} ) to delimit the nesting. This type of query generates a child rowset, which enables you to incorporate drill-down operations in the application.

### Example

The following hierarchical query produces a child rowset:

```
SELECT C_name,
       {SELECT O_orderkey,
          {SELECT L_partkey, L_linenumber
            FROM lineitem
            WHERE L_orderkey = O_orderkey} AS items
        FROM torder WHERE O_custkey=C_custkey} AS orders
FROM customer
```

The result has a three-tier hierarchical structure. The main (“root”) rowset has two columns (c\_name and orders, where orders is the first chapter {SELECT O\_orderkey ...}). The second, orders, column can be opened to display another (“child”) rowset. This child rowset includes the o\_orderkey column and *items*, a column that can be opened to display another child rowset:

c_name	orders		
John Friedman	[CHAPTER]		
Jack Jones	[CHAPTER]		
Ron Murphy	[CHAPTER]		
Richard Smith	[CHAPTER]		
Dan Hoffman	[CHAPTER]		
Chuck Rush	[CHAPTER]		
Al...	[CHAPTER]		
		o_orderkey	items
		7	[CHAPTER]
		39	[CHAPTER]
		71	[CHAPTER]
		102	[CHAPTER]
		128	[CHAPTER]
		228	[CHAPTER]
		257	[CHAPTER]
		323	[CHAPTER]
		l_partkey	l_linenumber
		12	1
		30	2

## Accessing Hierarchical Data Using SQL

Data stored hierarchically in a data source (such as information stored in arrays in RMS) can be referenced by using a `->` to denote the parent child relationship in the source:

```
FROM ... parent_name->child1->child2... [alias]
```

Or, using an alias for the parent table:

```
FROM ... parent_alias->child1->child2... [alias]
```

- ❖ The examples that follow assume hierarchical data with a parent employees rowset called *employees*. This rowset has ordinary columns plus a column called *hchild* (one row for each of the children of this employee). The child rowset *hchild* itself has (in addition to ordinary columns) a column called *hschool* (one row for each school attended by this child).

### Example

The following query retrieves information about children of **all** employees:

```
SELECT * FROM employees->hchild
```

Access to the *hchild* rowsets is via the parent rowset *employees*. Apart from this access, the employee data is not required by the query.

## Flattening Hierarchical Data Using SQL

You can produce a flattened view of hierarchical data by embedding a `SELECT` statement inside the list of columns to be retrieved by another `SELECT` statement. You use parentheses to delimit the nesting. This is equivalent to specifying a left outer join between the parent rowset and a child rowset, and the resulting rowset reproduces each row of the parent, combining it with one row for each of its children.

The nested `SELECT` statement can reference a child rowset (using the `parent->child` syntax) only in its `FROM` clause.

### Using an Alias

In order to list the hierarchical data with the parent data only, you must use an alias for the child data.

- ❖ Without an alias the query lists for each parent row all of the children of all of the parent rows.

Compare the following queries:

```
SELECT emp_id, (select name  
                from employee->hchild) from employee
```

And

```
SELECT emp_id, (select name
                from e->hchild)from employees e
```

The first query, without an alias, lists for each employee all of the children of all of the employees. The second query uses an alias and produces a list containing the children stored in an array called *hchild* belonging to each employee.

### Examples

The following query retrieves the number of children that study in each city for every city where the company has employees.

```
SELECT city, (SELECT COUNT(*)
              FROM employees->hchild->hschool A
              WHERE A.city = B.city)
FROM cities B
```

## Using Virtual Tables to Represent Hierarchical Data

Attunity Connect enables you to handle arrays and array-like structures (such a periodic group fields in Adabas) in non-relational data sources by converting the array data into a *virtual table* that resides in memory. The name assigned to the virtual table is a composite, consisting of the parent field name and the array name. For example, if a parent field is called *Employee* and the array is called *Empchild*, the virtual table is called *Employee\_Empchild*.

When arrays are nested within arrays, the virtual table name includes the names of all the parent fields and array names. For example, if a parent field is called *Employee* and an array called *Empchild* includes a nested array called *EmpChildSchools*, the virtual table name is called *Employee\_Empchild\_EmpChildSchools*.

A virtual table includes the following columns:

- The array fields.
- A column called *\_parent*, which contains the bookmark of the parent field. This column is generated automatically for the array and is read-only.
- A column called *\_rownum*, which identifies the row in the virtual table. This column is generated automatically for the array and is read-only.

The fields *\_parent* and *\_rownum* together uniquely identify each row in the virtual table. Use the *\_parent* field of the virtual table to identify the array field in the parent record when joining parent and child tables.

### Example

A record NATION stores data about countries for a mail order application and includes an array structure REGIONS. The array structure is converted into a virtual table NATIONS\_REGIONS. The following SQL accesses data from the virtual table NATIONS\_REGIONS:

```
SELECT * FROM NATION N, NATION_REGIONS R
WHERE R._PARENT = N.REGIONS
```

Once a virtual table has been created, it can be used in applications in the same way as any other table.

Note that this SQL statement accessing a virtual table is equivalent to the SQL statement using the -> syntax to access the array data in the table:

```
SELECT * FROM NATION->REGIONS
```

**Creating Virtual Tables** You generate virtual tables for arrays using an option in the Attunity Connect utility (NAV\_UTIL) that generates the metadata describing the array in the repository.

► **To create virtual tables:**

- Run the following command to generate the virtual tables for arrays defined in the parent table:

```
nav_util gen_array_tables ds_name table_name
```

Reference the virtual table in the query as you would reference any other table, using the name of the generated repository entry as the table name (this name is displayed when you run the utility).

## Copying Data From One Table to Another

Using the SQL SELECT statement in an INSERT statement enables data from one table to be copied into another table. The tables can be in different data sources, as long as the data types of data retrieved by the SELECT statement match the data types of the columns inserted in the table. For example:

```
insert into oracle:employees select * from disam:emp
```



## Passthru SQL

SQL UPDATE, INSERT, DELETE, DDL statements and SELECT statements can be passed directly to a relational data source, instead of being processed by the Attunity Connect Query Processor.

A data retrieval query can include joins where the data from one or more of the data sources is processed directly by the data source instead of the Attunity Connect Query Processor. This is particularly useful when dealing with a data source whose commands are not in standard SQL and whose data you want to join with data from other data sources.

### HP (Compaq) NonStop SQL/MP Platform

When specifying a passthru query to a HP (Compaq) NonStop SQL/MP database, if the query is not within a transaction, you must append the words "BROWSE ACCESS" at the end of the query.

For statements that do not return rowsets, Attunity Connect provides the following ways of passing SQL directly to the data source:

- For all SQL during a session from within the application via the connect string (see the specific application API (ODBC, etc. in *Attunity Connect Reference*).
- For a specific SQL statement from within the application (see below).

All SQL statements (both statements that do not return rowsets and statements that do return rowsets) can be passed as part of the SQL itself. This is particularly useful when dealing with a non-SQL data source, whose data you want to join with other SQL data (see page 121).

❖ Passthru queries are not supported for complex objects, such as BLOBs.

### Passthru Queries as Part of an SQL Statement

Both retrieval statements and non-returning rowset statements (such as DDL) can be passed directly to the data source as part of the SQL syntax.

To bypass the Attunity Connect Query Processor, include the query that you want to send directly to the data source in double braces (`{{query}}`), prefixed with the keyword `TEXT=`. Prefix all table names with the data source name specified in the binding configuration.

For retrieval queries, the passthru syntax is part of the FROM clause of a SELECT statement.

### Examples

- **A non-returning result:**

```
oracle:TEXT={{CREATE TABLE employee (emp_num integer  
NOT NULL, emp_name varchar(20))}}
```

- **As part of a SELECT statement:**

```
SELECT * FROM disam:nation,  
oracle:TEXT={{SELECT * FROM customer  
WHERE c_nationkey = n_nationkey  
AND c_custkey = ?}} (7)
```

where **disam** and **oracle** are data sources defined in the binding configuration. The query to the oracle database is passed directly to the database, bypassing the Attunity Connect Query Processor.

- ❖ Standard ANSI '92 SQL has been extended so that expressions involving columns of tables that appeared previously in the FROM list are used (from the nation table in the above example).

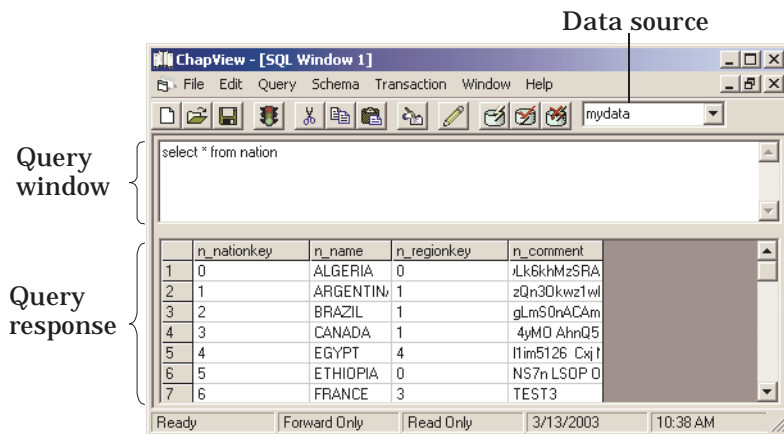
# Chapter 11 Using the Demo ADO Application

ChapView is an ADO application written in Microsoft VisualBasic, which demonstrates Attunity Connect's ability to:

- Connect to OLE DB and ODBC providers
- Execute queries that generate chapters
- Work with and modify chapters
- Execute parameterized queries
- Set recordset properties
- Work with schemas

You must have ADO version 2.5 installed to use the ChapView program.

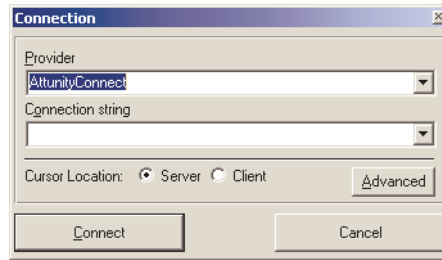
The following is an example of the graphical interface of the Attunity Connect ChapView program:



## Connecting to Attunity Connect via ChapView

► To connect to the Attunity Connect OLE DB Provider:

1. Invoke ChapView by selecting **Start | Programs | Attunity | Demos | Demo ADO Application**. This displays the Connection dialog box:

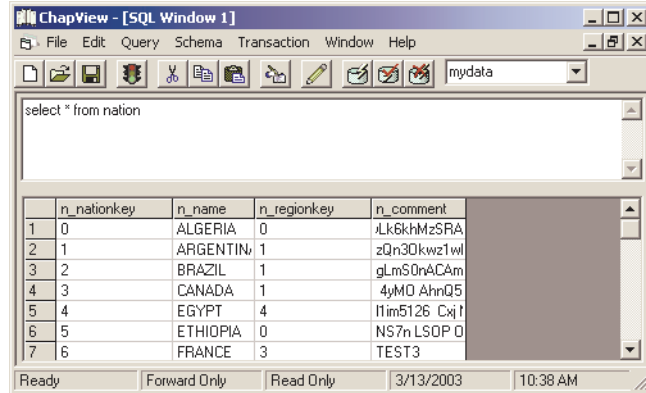


2. Note that ChapView automatically sets the provider for the connection. Enter any additional connection properties you want in the Connection string field or via the **Advanced** button.
  - ❖ If you change the binding configuration, via the **Advanced** button, the binding environment used is still the NAV binding environment.
3. The default cursor location is "Server". To work with a client-side cursor engine, select "Client".
4. Click **Connect** to connect to the provider.

## Specifying and Executing Queries

ChapView enables working with several query windows at the same time. To open a new query window, select **File | New** or press **<Ctrl>+<N>**.

As the following figure illustrates, the query window is divided into two parts, the upper part where you specify your SQL statement, and the lower area, which contains a grid with the resulting rowset.



► **To execute a query:**

- Select **Query | Execute**, or press <F5> or click the Execute button.
- ❖ You can specify a number of queries and then execute only one specific query by selecting the query to execute and pressing <F5>.

## Working with Parameterized Queries

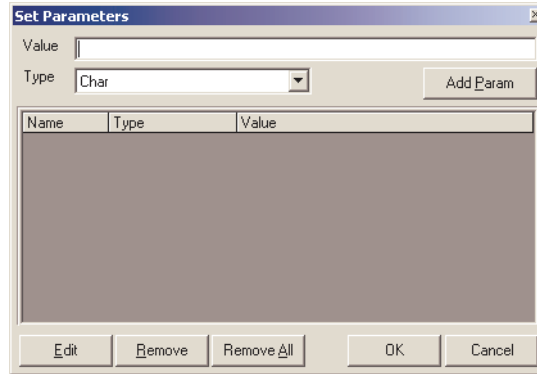
Attunity Connect enables you to specify parameterized queries, such as:

```
select * from nation where N_NATIONKEY between ? AND ?
```

Every query window in ChapView can have its own set of parameters.

► **To specify query parameters for the current window:**

1. Select Query | Parameters or press <Ctrl>+<P>. This displays the Set Parameters dialog box.



2. Specify the value of the parameter in the Value field.
3. Select the parameter type from the Type drop down list.
4. To change the parameter values, press <Edit> or double-click the parameter.
5. To remove a parameter, select its entry in the list and click **Remove**. To remove all parameters, click **Remove All**.
6. To set the parameter, click **OK**.


## Modifying Data in a Recordset

ChapView enables you to add, update and delete data in a resulting recordset, if the lock type is not "Read Only".

ChapView also supports transactions. The following transaction-related methods are provided in the Transaction menu:

- Begin – starts a transaction.
- Commit – commits a transaction.
- Rollback – rolls back a transaction.

► **To modify a resulting recordset:**

1. Click the modify button  in the toolbar. This displays the modify toolbar in place of the edit box of the SQL.
2. Double-click on the row whose value you want to modify. When you begin modifying a certain row, you can click Update or Cancel Update, to perform or cancel the modifications, respectively.

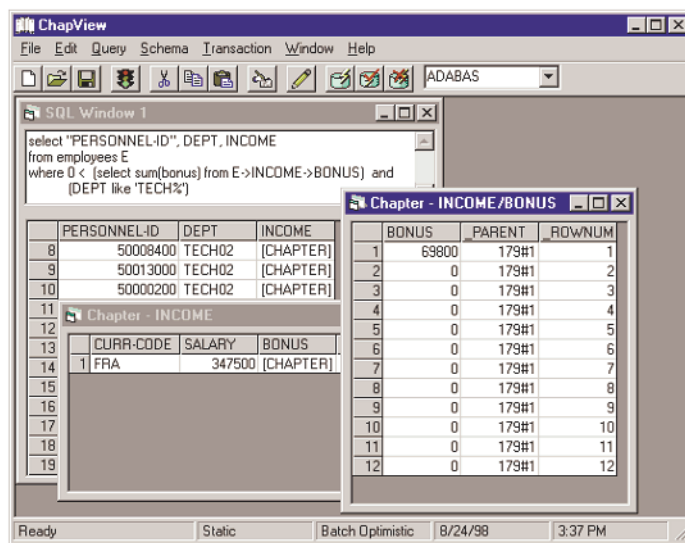
- ❖ If you are working with BatchOptimistic locking, you must also click on UpdateBatch or CancelBatch to confirm the batch update or cancel it.

In Modify mode, you can also add or delete an existing row.

To exit Modify mode, click **Close**.

## Working with Chapters

ChapView enables you to view and modify chapters. To view a particular chapter, double-click the grid cell containing the chapter.



## Modifying Chapters

With ChapView you can modify chapters. Note that in ADO 1.5 the chapter is always opened with BatchOptimistic mode, which means you must apply UpdateBatch or CancelBatch after you modify the recordset.

To modify a chapter, click on the modify icon on the toolbar and apply modifications as you do for a normal recordset.

Modify chapters carefully; you may end up with several open chapters. Sometimes you may need to close and open the chapter or requery the parent to see the changes.

## Specifying Recordset Properties

ChapView enables you specify the following recordset properties:

- Cursor type
- Lock type
- Cache size
- Maximum number of records retrieved

► **To set the recordset properties:**

1. Select **Query | Recordset Properties** or press <F4> to display the Recordset Properties dialog box:



2. Specify the value of the fields corresponding to the recordset property you want to set:

**Cursor Type** – Specifies one of the following types of cursor:

- ForwardOnly
- Keyset
- Dynamic
- Static

The default is ForwardOnly, meaning you can only scroll forward in an ADO recordset (note that ChapView caches the rowset to the Grid).

For Attunity Connect your choices are "Forward only" and "Static".

**Lock Type** – Specifies one of the following locking types:

- Read only
- Pessimistic
- Optimistic
- BatchOptimistic

The default is read only.



**Cache size** – Specifies the size of the cache for the recordset. The default is -1, which sets ChapView to use the ADO default for the cache size of the recordset.

**MaxRecord** – Specifies the maximum number of records that are retrieved. Setting -1, indicates that all rows are returned.

- ❖ If you select the Save changes check box, the changes you make to the properties are saved in the registry (and will serve as new defaults).

## Working with Schemas

ChapView enables you to view ADO schemas. The following Schema type are supported:

- Catalogs
- Columns
- Foreign Keys
- Indexes
- Primary Keys
- Procedures
- Procedure Columns
- Provider Types
- Statistics
- Tables

### ► To view a specific schema:

- Select the schema type from the "Schema" menu. The following example displays the Tables schema of the mydata data source:

	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	COLUMN_C
1	mydata	[NULL]	CORSECTN	CRSCOURS	[NULL]
2	mydata	[NULL]	CORSECTN	CRSBEG	[NULL]
3	mydata	[NULL]	CORSECTN	CRSECTN	[NULL]
4	mydata	[NULL]	CORSECTN	CRSDAY	[NULL]
5	mydata	[NULL]	CORSECTN	CRSEKEY	[NULL]
6	mydata	[NULL]	MYDATA	midinitial	[NULL]
7	mydata	[NULL]	MYDATA	firstname	[NULL]
8	mydata	[NULL]	MYDATA	id	[NULL]
9	mydata	[NULL]	MYDATA	lastname	[NULL]
10	mydata	[NULL]	MYDATA1	id	[NULL]
11	mydata	[NULL]	MYDATA1	lastname	[NULL]

You can also copy and paste, from a schema grid to the query text box in the query window.



# Appendix A Importing Metadata for a Data Source

Metadata for Enscribe, IMS/DB, RMS and VSAM can be imported via the Metadata Import perspective of Attunity Studio. Metadata for other data sources can be generated from COBOL copybooks, also within Attunity Studio.

A number of stand-alone import utilities also exist to generate Attunity Connect metadata. For details, refer to *Attunity Connect Reference*.

Additionally, metadata that can be extracted from COBOL copybooks can be imported to Attunity Connect using the Attunity Studio Metadata Import perspective.

The following describes the flow when importing metadata for a VSAM data source. Importing metadata for any data source is similar.

## ► To import metadata:

1. In the Configuration perspective, right-click the data source for which you want to import the metadata and select Open import perspective from the popup menu.

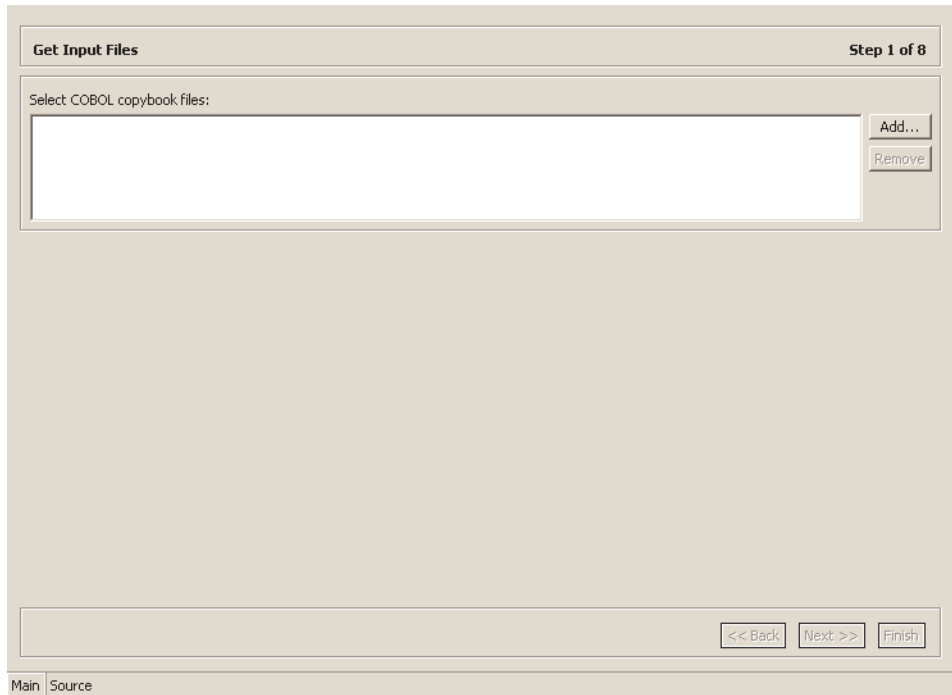
The Metadata Import perspective opens with the selected data source displayed in the explorer tree.

2. Right-click the data source and select Open import perspective.

The Import perspective opens with the data source appearing in the in the explorer tree.

3. Right-click the data source and choose New Import.
4. Enter a name for the import. The name can contain letters and numbers and the underscore character only.
5. Click **Finish**.

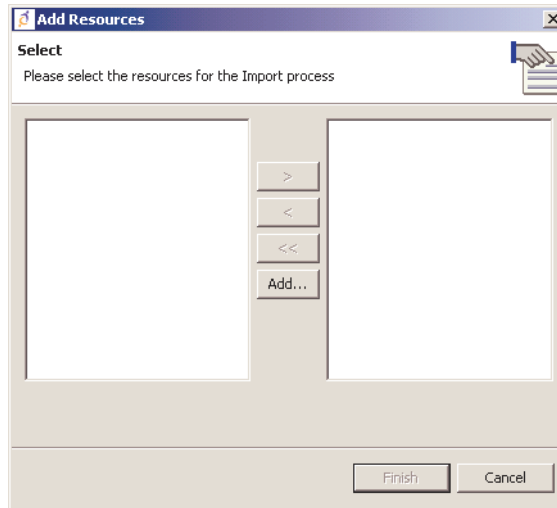
The metadata import wizard opens in Attunity Studio.



COBOL copybooks are used to create the metadata.

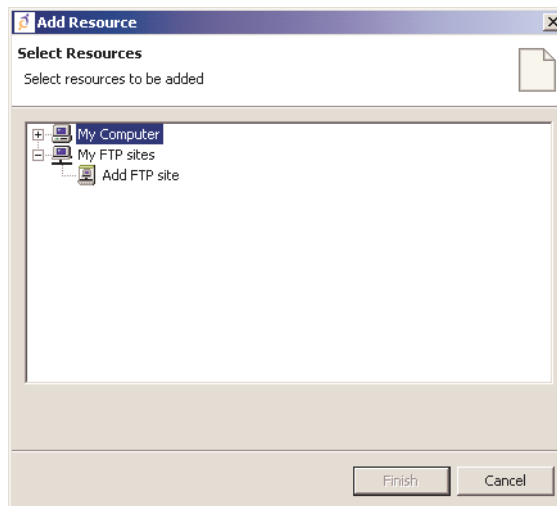
6. Click **Add** to add the COBOL copybooks defining the data.

A window opens which lists the files to use in the import.



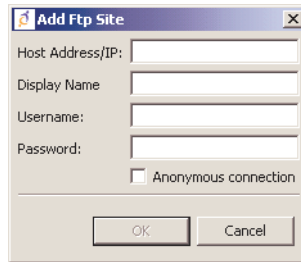
7. Click **Add** to add the files to use during the import.

The Add Resource window opens, which provides the option to select files from the local machine or FTP the files from another machine.



8. If the files are local, expand My Computer to select the files from the appropriate directory.

If the files are on another machine, double click Add FTP site to access the machine where the COBOL files are located.



9. Enter the name of the server where the files reside and, if not using anonymous access, enter a username and password to access the machine.

Click **OK**. The machine is displayed in the My FTP sites tree, showing the high-level qualifier for the user.

To change the high-level qualifier, right-click the machine and choose Change Root Directory from the popup menu. Enter the root directory and click **OK**.

10. Right-click the machine and choose Set Transfer Type from the popup menu. Enter the transfer type (ASCII or BINARY) and click **OK**.
11. Navigate to the location where the files reside and select the files to be transferred.
12. Click **Finish**.

The Select window is redisplayed with the list of transferred files.

13. Make sure that the files you want to use for the import are displayed in the right pane and click **Finish**.

The format of the COBOL copybooks must be the same. That is, you cannot import a COBOL copybook that uses the first six columns with a COBOL copybook that ignores the first six columns. In this case, you must do separate imports.

❖ You can import the metadata from one COBOL copybook and later add to this metadata by redoing the import using different COBOL copybooks.

14. Click **Next**.

## 15. Apply filters to the copybooks.

Apply Filters Step 2 of 8

Click Next to analyze and convert the source files. You may also change various filter options.

Property	Value
<input type="checkbox"/> cobolFilter	
<input type="checkbox"/> compilerSourceSettings	
COMP_6 switch (for MICROFOCUS compil...	COMP-6'2'
Compiler source	Default / Not known / Other
Storage mode (for MICROFOCUS compiler...	NOIBMCOMP
Ignore first 6 characters	false
Ignore from character 72	true
Prefix nested columns	true
Replace hyphens (-) in record and field name...	true
<input type="checkbox"/> template	
Find	
Ignore case sensitivity	true
Replace with	

Main | Source

The following filters are available:

**COMP\_6 Switch** – The MicroFocus COMP-6 compiler directive. Specify either COMP-6'1' to treat COMP-6 as a COMP data type or COMP-6'2' to treat COMP-6 as a COMP-3 data type.

**Compiler Source** – The compiler vendor.

**Storage Mode** – The MicroFocus Integer Storage Mode. Specify either NOIBMCOMP for byte storage mode or IBMCOMP is for word storage mode.

**ignoreFirst6** – Ignore the first six columns in the COBOL copybook.

**ignoreFrom72** – Ignore columns 73 to 80 in the COBOL copybook.

**prefixNestedColumns** – Prefix all nested columns with the previous level heading.

In addition, you can specify a search string and the string that will replace this search string in the generated metadata, and whether the replacement is dependent on the case of the found string:

**Find** – Searches for the specified value.

**Ignore Case Sensitivity** – Specifies whether to be sensitive to the search string case.

**Replace with** – Replaces the value specified for Find with the value specified here.

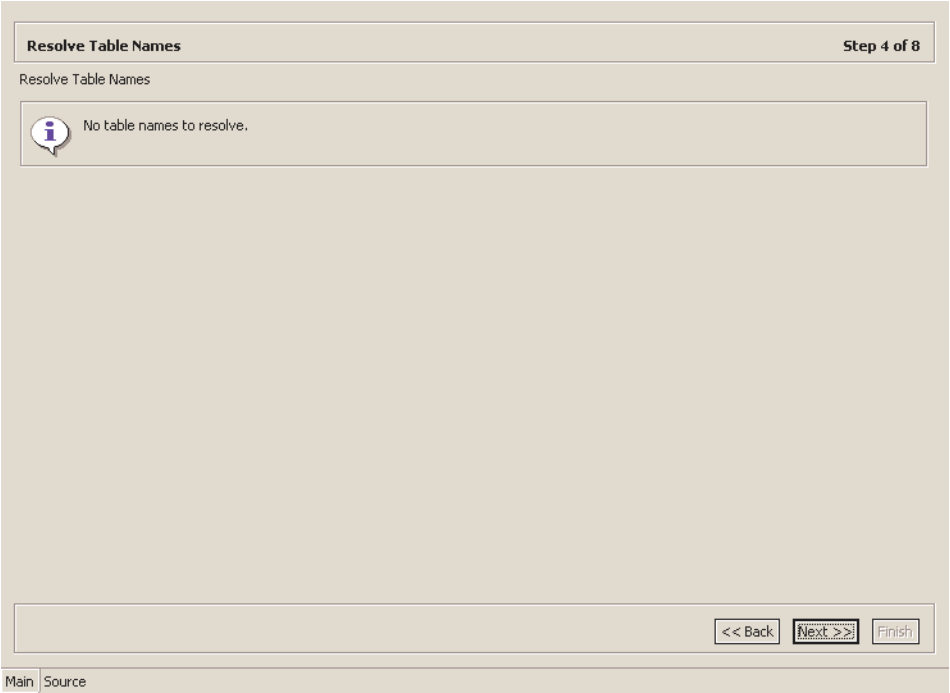
- Click **Next** to apply the filter settings.
- The records that are identified in the COBOL copybooks are displayed. You can select the records you want to import.

[illegible]

18. Select the records to import and click **Next**.



The import manager identifies the names of the records in the COBOL copybooks that will be imported as tables. If the names are unique, the following window is displayed:





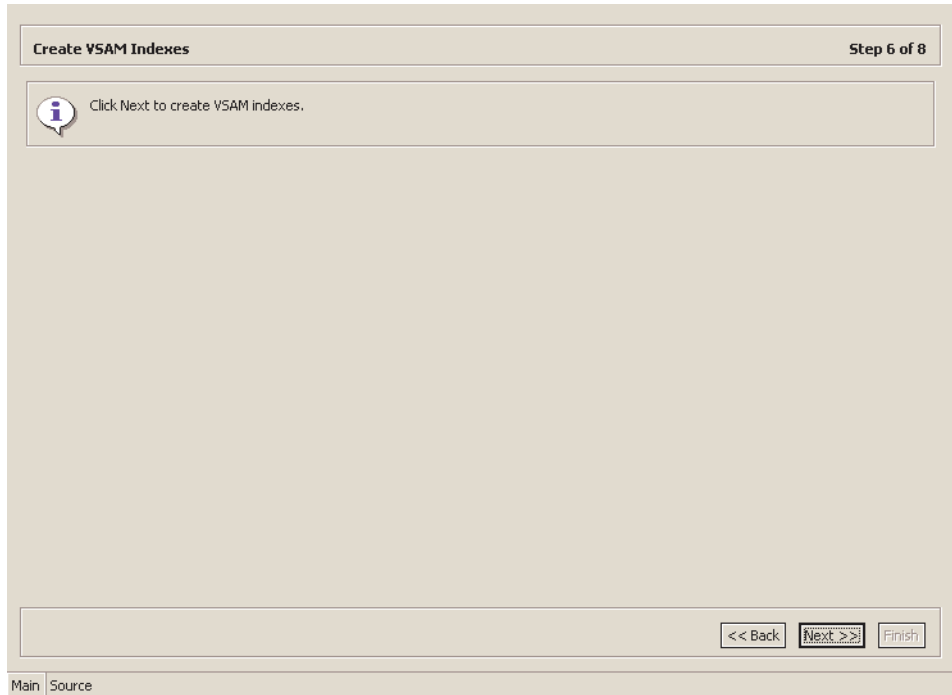
If the VSAMCICS Import Manager is used, specify both physical file name and the logical file name for each record listed.

[illegible]

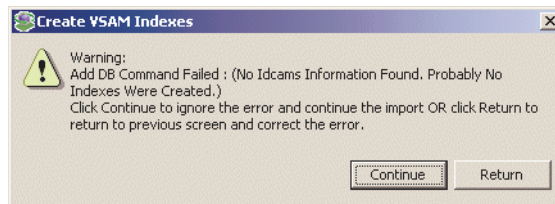
If the VSAM Import Manager is used, specify the physical file name for each record listed, including the high-level qualifiers.

21. Click **Next**.

22. The machine must be accessed to retrieve information to build metadata about the indexes.

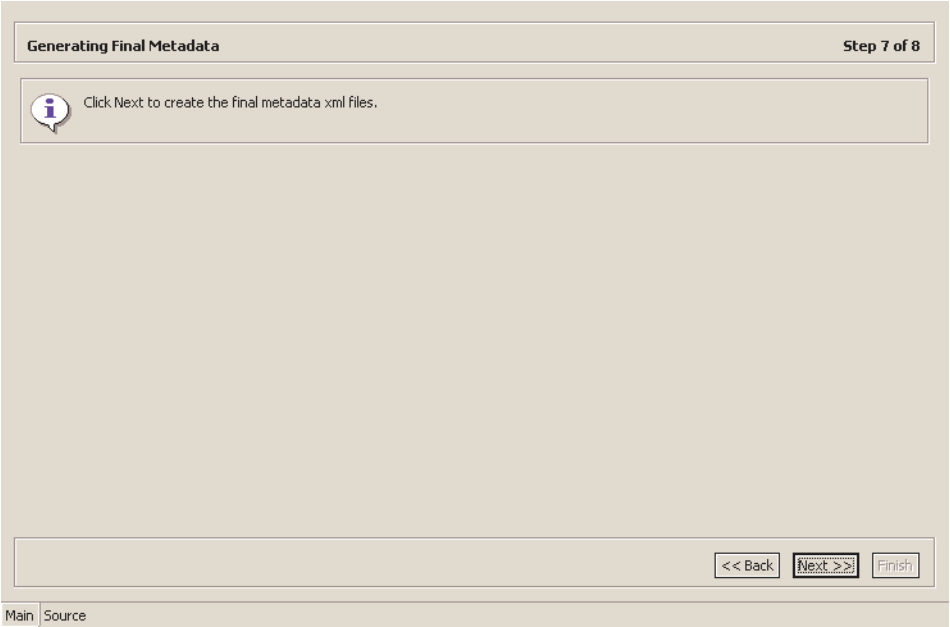


If you try to build the indexes and the step fails for any reason (such as the machine is temporarily not accessible), a warning message similar to the following message is issued and you continue to the next step.



23. Click **Continue** to go to the next step.

24. Click **Next** to generate the metadata within Attunity Studio.



25. Specify that you want to transfer the metadata to the server machine and click **Finish**.

**Import Metadata** Step 8 of 8

Click Finish to end the import process.

**Deploy Metadata:**  
Do you want to import the metadata to the server?

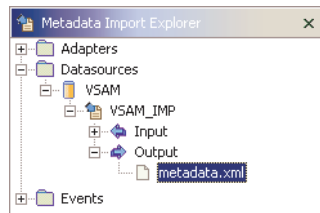
☒ No  
☐ Yes

<< Back   Next >>   Finish

Main | Source

The metadata is imported based on the options specified and is stored on the machine in the Explorer containing the data source. An XML representation of the metadata is generated. This XML file can be viewed by expanding the Output node.

After performing the import, you can view the metadata in the Metadata perspective. You can also make any fine adjustments to the metadata and maintain it, as necessary.



# Appendix B Importing Metadata for an Adapter

When COBOL Copybooks defining the adapter are available, metadata for the adapter can be imported to Attunity Connect. Otherwise, metadata is defined manually as described in "Metadata for an Application Adapter" on page 79.

Metadata is imported and saved using the Metadata Import perspective. Additionally, metadata that can be extracted from COBOL copybooks can be imported to Attunity Connect using the Attunity Studio Metadata Import perspective.

The following describes the flow when importing metadata for a CICS adapter. Importing metadata for any adapter is similar.

► **To import metadata:**

1. In the Configuration perspective, right-click the adapter for which you want to import the metadata and select Open import perspective from the popup menu.

The Metadata Import perspective opens with the selected adapter displayed in the explorer tree.

2. Right-click the data source and select Open import perspective.

The Import perspective opens with the data source appearing in the in the explorer tree.

3. Right-click the data source and choose New Import.
4. Enter a name for the import. The name can contain letters and numbers and the underscore character only.
5. Click **Finish**.

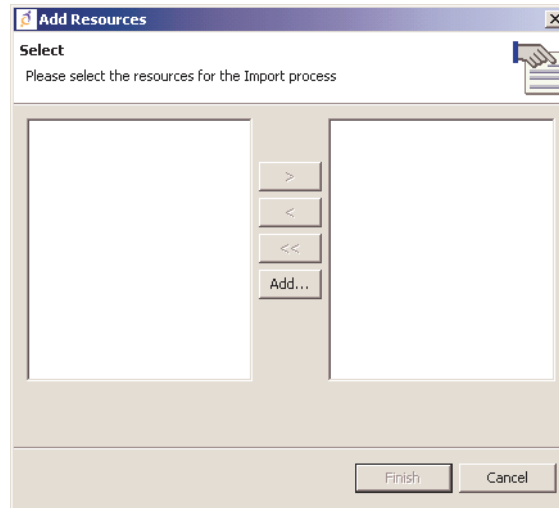
The metadata import wizard opens in Attunity Studio. for the CICS import, COBOL copybooks are used to create the metadata.

The screenshot shows a wizard window titled "Get Input Files" with a progress indicator "Step 1 of 5". Inside the window, there is a label "Select COBOL copybook files:" above a large empty rectangular box. To the right of this box are two buttons: "Add..." and "Remove". At the bottom right of the window are three buttons: "<< Back", "Next >>", and "Finish". At the bottom left, there are two tabs: "Main" and "Source".

6. Click **Add** to add COBOL copybooks.

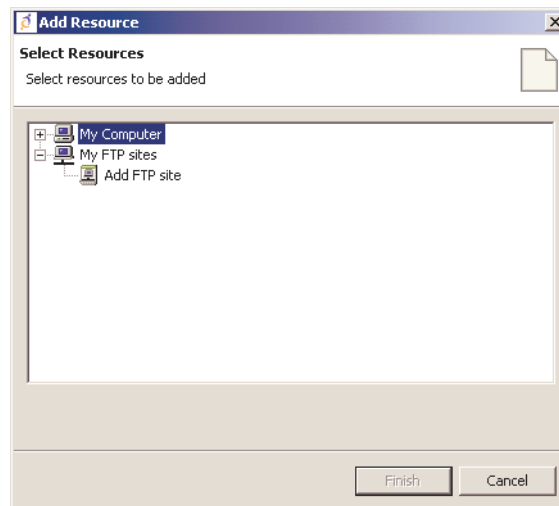


A window opens which lists the files to use in the import.



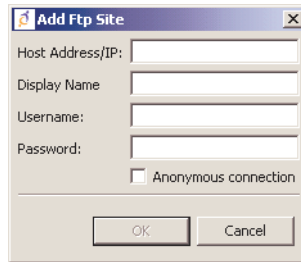
7. Click **Add** to add the files to use during the import.

The Add Resource window opens, which provides the option to select files from the local machine or FTP the files from another machine.



8. If the files are local, expand My Computer to select the files from the appropriate directory.

If the files are on another machine, double click Add FTP site to access the machine where the COBOL files are located.



9. Enter the name of the server where the files reside and, if not using anonymous access, enter a username and password to access the machine.

Click **OK**. The machine is displayed in the My FTP sites tree, showing the high-level qualifier for the user.

To change the high-level qualifier, right-click the machine and choose Change Root Directory from the popup menu. Enter the root directory and click **OK**.

10. Right-click the machine and choose Set Transfer Type from the popup menu. Enter the transfer type (ASCII or BINARY) and click **OK**.
11. Navigate to the location where the files reside and select the files to be transferred.
12. Click **Finish**.

The Select window is redisplayed with the list of transferred files.

13. Make sure that the files you want to use for the import are displayed in the right pane and click **Finish**.

The format of the COBOL copybooks must be the same. That is, you cannot import a COBOL copybook that uses the first six columns with a COBOL copybook that ignores the first six columns. In this case, you must do separate imports.

- ❖ You can import the metadata from one COBOL copybook and later add to this metadata by redoing the import using different COBOL copybooks.

**Get Input Files** Step 1 of 5

Select COBOL copybook files:

- COBCPL
- COBDEMO
- MATHSAMP

Add...  
Remove

<< Back Next >> Finish

Main | Source

14. Click **Next**.

## 15. Apply filters to the copybooks.

Apply Filters Step 2 of 8

Click Next to analyze and convert the source files. You may also change various filter options.

Property	Value
<input type="checkbox"/> cobolFilter	
<input type="checkbox"/> compilerSourceSettings	
COMP_6 switch (for MICROFOCUS compil...	COMP-6'2'
Compiler source	Default / Not known / Other
Storage mode (for MICROFOCUS compiler...	NOIBMCOMP
Ignore first 6 characters	false
Ignore from character 72	true
Prefix nested columns	true
Replace hyphens (-) in record and field name...	true
<input type="checkbox"/> template	
Find	
Ignore case sensitivity	true
Replace with	

Main | Source

The following filters are available:

**COMP\_6 Switch** – The MicroFocus COMP-6 compiler directive. Specify either COMP-6'1' to treat COMP-6 as a COMP data type or COMP-6'2' to treat COMP-6 as a COMP-3 data type.

**Compiler Source** – The compiler vendor.

**Storage Mode** – The MicroFocus Integer Storage Mode. Specify either NOIBMCOMP for byte storage mode or IBMCOMP is for word storage mode.

**ignoreFirst6** – Ignore the first six columns in the COBOL copybook.

**ignoreFrom72** – Ignore columns 73 to 80 in the COBOL copybook.

**prefixNestedColumns** – Prefix all nested columns with the previous level heading.

In addition, you can specify a search string and the string that will replace this search string in the generated metadata, and whether the replacement is dependent on the case of the found string:

**Find** – Searches for the specified value.

**Ignore Case Sensitivity** – Specifies whether to be sensitive to the search string case.

**Replace with** – Replaces the value specified for Find with the value specified here.

17. Click **Add** to add an interaction for the CICS adapter.

**Interaction name** – The name of the interaction.

- **sync-receive** – The interaction

- Input** – Identifies an input record. The input record is the data structure for the outbound interaction. The records generated from the COBOL programs specified at the beginning of the procedure are listed in a drop-down list. Select the relevant interaction record.

If the interaction does not require an input record, the record specified here is ignored.

**Output** – Identifies an output record. The output record is the data structure for the results of the outbound interaction. The records generated from the COBOL programs specified at the beginning of the procedure are listed in a drop-down list. Select the relevant record for the interaction.

- ❖ You must specify an output record for the interaction if the mode is set to sync-send-receive or sync-receive, before you can click **Next**.

**Description** – Free text describing the interaction.

**Interaction-Specific Parameters** – The name of the program that is called to execute the interaction.

- ❖ You must specify a program name for each interaction.

19. Add as many interactions as necessary and then click **Next**.
20. Click **Next** to generate the metadata definitions for the adapter.
21. Specify that you want to transfer the metadata to the server machine and click **Finish**.

The metadata is imported based on the options specified and is stored on the machine with the adapter. An XML representation of the metadata is generated. This XML file can be viewed by expanding the Output node.

After performing the import, you can view the metadata in the Metadata perspective. You can also make any fine adjustments to the metadata and maintain it, as necessary.

## Appendix C Generating Metadata for a Database Adapter

You define metadata (an adapter definition) for the database adapter, in which you specify the SQL statements you want executed.

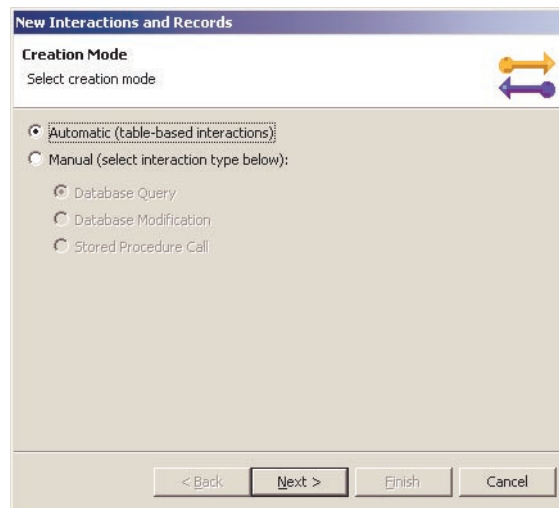
You can add tables from any of the data sources in the same binding as the Database adapter and that you want to access with the interaction.

- ❖ The Database adapter requires predefined SQL, as opposed to the Query adapter, where the SQL is specified at runtime.

### ► To generate metadata:

1. Right-click the Database adapter and choose **Edit metadata** to open the Metadata perspective, with the adapter displayed under the adapters list.
2. In the Metadata perspective, right-click the Interactions node and choose **New** to open the New Interaction wizard.

The New Interaction wizard opens with two options displayed, to create interactions:



**Automatic** – Four interactions are generated for each VSAM table enabling the following SQL to be executed:

- SELECT
- INSERT
- UPDATE
- DELETE

**Manual** – One interaction is generated, based on the type of SQL selected:

- Database Query (a SELECT statement)
- Database Modification (an INSERT, UPDATE or DELETE statement)
- Stored Procedure Call.

► **To automatically create interactions:**

The following steps describe creating interactions using the automatic option. Also see "To manually create interactions:" on page 155.

1. Click **Next**.

The Select Tables window opens, enabling you to add tables from the data source that you want to access with the interaction.

**New Interactions and Records**

**Select Tables**  
Select tables for new interactions

DataSource	Table	Key Columns	Batch Record Count

☐ Use XML Field ☐ Use Key in Select ☐ Upsert

2. Click **Add** to add tables.





You can select a table and set the following:

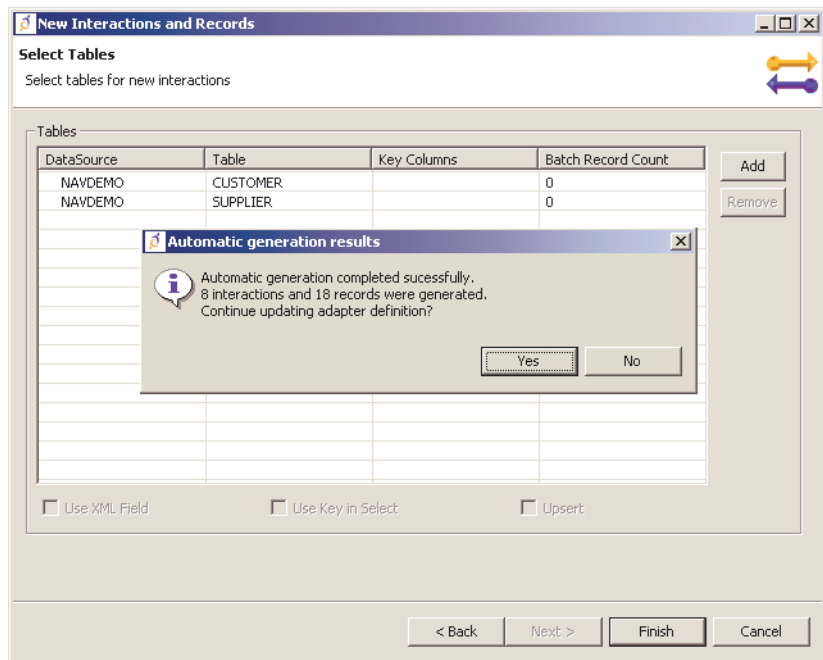
**Use XML Field** – The output is formatted as XML according to its actual structure. This is useful for tables which include variant fields and arrays.

- ❖ For this field to be available, the `exposeXmlField` in the misc section of the binding environment properties must be set to true.

**Use Key in Select** – The key column in the table is used in the SELECT statement.

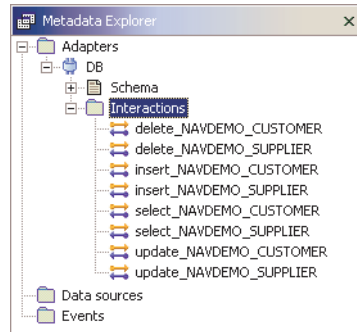
**Upsert** – When inserting a row, if the row doesn't exist, it is inserted. Otherwise, the row is updated with the new information.

4. Click **Finish**. Four interactions are generated for each table selected, together with the record structures to support the interactions and the responses from the data sources.



- ❖ If the Upsert property is checked, only three interactions are generated. An insert interaction is generated with an update property instead of both an insert and update interaction.

5. Click **Yes** to complete the task. The interactions and the record structures that relate to the interactions are displayed in the Metadata perspective.



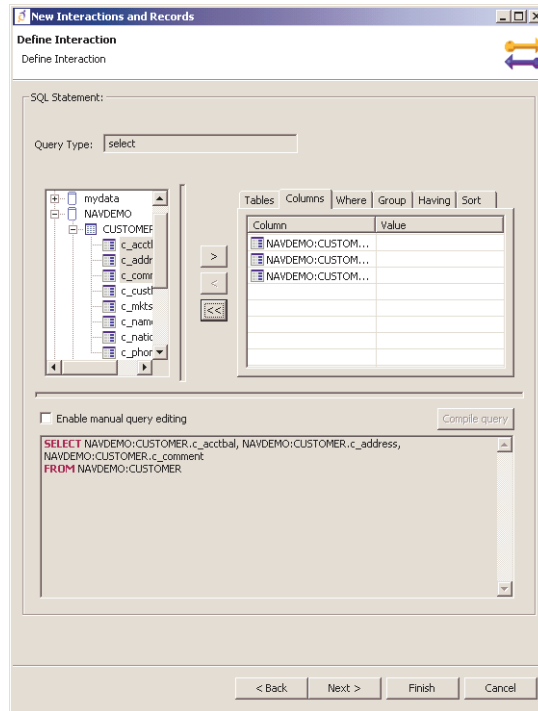
► **To manually create interactions:**

The following steps describe creating interactions using the manual option. Also see "To automatically create interactions:" on page 152.

1. Select the type of SQL (Query, Modification or stored procedure) and click **Next**.

A window is displayed where you provide a name for the interaction. For a query or modification you can also select an existing query as the basis for the interaction.

2. Click **Next**. A window is displayed where you can build the query.



The query is built as follows:

### Selecting tables

- In the left pane, expand the data source where table resides.
- Select the table and click the right button to move the table to the right pane of selected tables.

### Selecting Columns

- In the left pane, expand the data source and the table for the column.
- Open the Columns tab in the right pane.
- Select the column and click the right button to move the column to the right pane.

### Joining columns from different tables

When a column having the same name as another column selected from a different table is selected, the Create Joint tables pane opens.

- Expand the table and select the column you want to join.

- Click the right arrow to move the column to the right pane.
- Optionally, click next and edit the join statement.
- Click **Finish**.

#### Adding conditions in a WHERE clause

WHERE clauses are set in the Where tab.

- Select and move the column you are setting the WHERE clause for to the right pane.
- Set the operator and value conditions as needed.

#### Grouping Columns

Columns are grouped in the Group tab.

- Select and move the columns you are grouping to the right pane.

#### Filtering results using a HAVING clause

The HAVING clause provides conditions for grouping columns.

HAVING clauses are set in the Having tab.

- Select and move the column you are filtering to the right pane.
- Set the operator and value conditions as needed.

#### Sorting results

Query results are sorted in the Sort tab.

- Select and move the column whose query result you want to sort to the right pane.
  - Select the sorting order as either ascending or descending.
3. Click **Next**. A window is displayed where you specify interaction properties. Refer to *Attunity Connect Reference* for details.
  4. Click **Next**. A window is displayed where you specify input parameters for the interaction. Refer to *Attunity Connect Reference* for details.
  5. Click **Finish** to generate the interaction, including the record schema required to support the interaction input and output.

## Modifying the Interactions

You can modify the interaction definitions to the exact requirements of the application in the Metadata perspective. The following example uses a DELETE interaction to describe how the interactions can be modified.

1. In the Metadata perspective, right-click the interaction to modify and choose Edit Metadata from the popup menu.

The adapter metadata editor opens, displaying the Interaction General tab.

The screenshot shows the 'Adapter Metadata Editor' window. The 'Details' section at the top contains the following fields:

- Interaction name:** delete\_NAVDEMO\_CUSTOMER
- Description:** (empty text box)
- Mode:** sync-send-receive (selected from a dropdown menu)

The 'Input/Output Definitions' section below contains:

- Input record:** delete\_NAVDEMO\_CUSTOMER (with a 'View ...' button)
- Output record:** delete\_NAVDEMO\_CUSTOMERResponse (with a 'View ...' button)

At the bottom, a tabbed interface shows the following tabs: General, **Interaction General** (highlighted with a red circle), Interaction Advanced, Schema General, Schema Record, and Source.

The Interaction General tab displays general information about the way the interaction is executed. You can add a description of the interaction and define the mode of operation for the interaction.

The following modes are available:

- **sync-send-receive** – The interaction sends a request and expects to receive a response.
  - **sync-send** – The interaction sends a request and does not expect to receive a response.
  - **sync-receive** – The interaction expects to receive a response.
- The information for a request is passed in the input record. The information for the response from the data sources is passed in the output record.

2. Click the Interaction Advanced tab to display specific information about the interaction.

Interaction Type: Database Modification

DB2  
disam  
mydata  
MyDisam  
NAVDEMO  
SYS

Tables Columns Where Group Having Sort

Table	Alias

☐ Enable manual query editing Compile query

**UPDATE** CUSTOMER SET c\_custkey = :c\_custkey, c\_name = :c\_name, c\_address = :c\_address, c\_nationkey = :c\_nationkey, c\_phone = :c\_phone, c\_acctbal =

Interaction Properties  
☐ Pass Through ☐ Keep Alive ☐ Fail on Zero Affected

Parameters

Name	Type	Nullable	Default	Context F...	Bind To Sqls
c_custkey	number	false			
c_name	string	false			
c_address	string	false			
c_nationkey	number	false			
c_phone	string	false			
c_acctbal	number	false			
c_mktsegment	string	false			
c_comment	string				
c_custkeyKey	number	false			
c_nameKey	string	false			
c_addressKey	string	false			
c_nationkeyKey	number	false			
c_phoneKey	string	false			
c_acctbalKey	number	false			
c_mktsegment...	string	false			
c_commentKey	string				

Add  
Delete

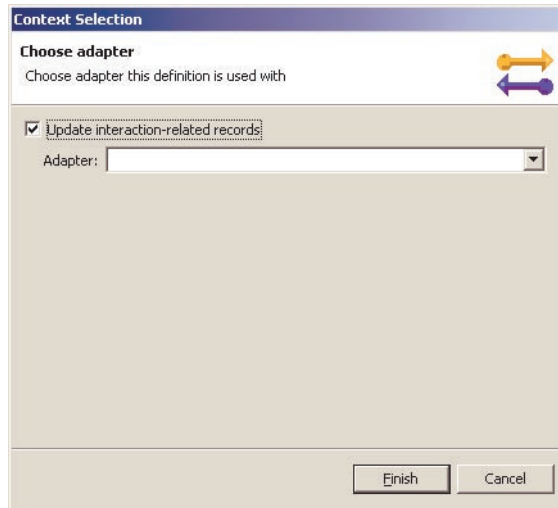
General | Interaction General | **Interaction Advanced** | Schema General | Source

As, required, change the SQL and the parameters associated with the SQL. The SQL is changed as described above in "To manually create interactions:" on page 155.

Parameters are specified in a SET clause or in a WHERE clause with the following format:

:parameter\_name

3. Dependent on the changes made to the SQL, when you close the editor, or click the **Save** button, the Context Selection window opens.



Specify the adapter from the list and check the **Update interaction-related records** checkbox. Any changes that need making to the record structures in the schema part of the metadata are done automatically.

- ❖ The interaction records are built based on all the fields in the table and cannot be changed manually, even if you change the SQL so that less fields are involved.



# Index

## A

- access
  - through a firewall 102
- accessing data
  - checking a connection 107
  - multiple databases 116
  - via ADO 111
  - via JDBC 107
  - via ODBC 108
- ACX
  - connect verb 111
  - disconnect verb 112
  - execute verb 112
- ADO
  - batching SQL 116
  - connect string 111
- architecture
  - client/server 19, 47
  - daemon 19
  - database drivers 14
- arrays
  - creating virtual tables 120
  - flattening hierarchical data 118
- Attunity Connect
  - overview 9
- Attunity Repository 18

## B

- batching SQL 115
- binding configuration
  - adding 36
  - link to user profile 99

## C

- chapters 116
- client/server architecture 19
- client-server architecture 47
- communications
  - encrypting 103
- connect string
  - ADO 111
  - DSN 108

- JDBC 107
- ODBC 108
- connect verb 111
- connecting
  - checking a connection 107
  - connect verb 111
  - via ADO 111
  - via JDBC 107
  - via ODBC 108
- connection
  - destroying with disconnect 112
- connection information
  - multiple databases 116

## D

- daemon
  - adding 36
  - architecture 19
  - configuring server logging 64
  - configuring the log 62
  - server modes 56
  - shutting down 54
  - starting 49
  - workspace administration rights 98
  - workspaces 47
- database driver
  - architecture 14
- databases
  - supported 11, 15
- disconnect verb 112
- drivers
  - checking a connection 107
- DSN
  - connect string 108

## E

- encrypting network communications 103
- execute verb 112

## F

- firewall
  - access through 102

## H

- hierarchical data
  - creating virtual tables 120
- hierarchical queries 116
  - flattening hierarchical data 118

## I

- interactions
  - executing 112

## J

- JDBC
  - connect string 107
- joins
  - multiple databases 116

## L

- log files
  - configuring for daemon 62
  - server log 64

## M

- machines
  - adding 36
- multiple databases
  - joining 116
- multiple results API 115

## N

- network communications
  - encrypting 103

## O

- ODBC
  - connect string 108
- overview 9

## P

- passthru SQL 121
  - via SQL syntax 121
- platforms
  - supported 21
- prestarted servers 59

- configuring 59, 60
- prestarted sub-tasks 60

## S

- security
  - access through firewalls 102
  - access to a remote machine 101
  - workspace administration rights 98
- SELECT statement
  - passthru SQL 121
- server
  - configuring logging 64
- server modes
  - configuring 56
- servers
  - prestarted servers 59, 60
  - reusable 58
  - ReuseLimit daemon parameter 58
  - threads 57
- SQL
  - batching queries 115
  - flattening hierarchical data 118
  - hierarchical queries 116
  - joining multiple databases 116
  - multiple results API 115
  - passthru queries 121
  - passthru SQL syntax 121
- string URL (JDBC) 107

## T

- threads
  - configuring 57

## U

- UNIX
  - file system 62, 64
- user profile
  - adding 37
  - link to binding configuration 99

## V

- virtual tables
  - creating 120

**X**

## XML

connect verb 111

disconnect verb 112

execute verb 112

